Multi-channel proximity estimation to improve privacy preserving contact tracing methods

Master thesis

Author Eric Lanfer
elanfer@uni-osnabrueck.de

Examiners Prof. Dr. rer. nat. Nils Aschenbruck
Prof. Dr. ir. Roland van Rijswijk-Deij

Institution Institute of Computer Science
Osnabrück University

Design and Analysis of Communication Systems Group
University of Twente, Enschede

Date May 25, 2021

Abstract

In the wake of the COVID 19 pandemic that swept through society in 2020, there has been a strong emphasis on the use of smartphone-based proximity tracing systems. Many of the systems used Bluetooth Low Energy (BLE) signals to estimate the distance between two people. This is a method whose quality depends on many factors and, therefore, often delivers inaccurate results. In this work, we present a multi-channel approach to improve proximity estimation. We have developed and evaluated a combined classification based on BLE and IEEE 802.11 signals. Our approach is able to improve the distance estimation significantly; in the evaluation on our ground truth data, the method had an f1-score of more than 0.9. Furthermore, we were able to achieve good results with our approach in everyday public transport scenarios. However, in our first implementation based on IEEE 802.11 probe requests, we also encountered privacy problems and limitations in situations with strongly changing signal propagation. These problems have been analysed in detail, and we are making recommendations for future work to enable using our approach in everyday life.

Zusammenfassung

Ausgelöst durch die COVID-19-Pandemie, die im Jahr 2020 über die Gesellschaft hereinbrach, wurde ein starker Fokus auf die Verwendung von Smartphone-basierten Proximity-Tracing-Systemen gelegt. Viele dieser Systeme verwendeten BLE-Signale, um die Entfernung zwischen zwei Personen zu schätzen. Dies ist eine Methode, deren Qualität von vielen Faktoren abhängig ist und daher oft ungenaue Ergebnisse liefert. In dieser Arbeit wird ein multi-channel Ansatz vorgestellt, um die Abstandsschätzung zu verbessern. Es wurde eine kombinierte Klassifizierung auf Basis von BLE und IEEE 802.11-Signalen entwickelt und evaluiert. Der Ansatz ist in der Lage, die Abstandsschätzung deutlich zu verbessern; in der Auswertung auf den Ground-Truth-Daten hatte die Methode einen f1-Score von mehr als $0,9$. Außerdem konnten mit dem Ansatz in alltäglichen Szenarien des öffentlichen Nahverkehrs gute Ergebnisse erzielt werden. Bei einer ersten Implementierung auf Basis von IEEE 802.11 probe requests, wurden jedoch auch Datenschutzprobleme und auf Einschränkungen in Situationen mit stark veränderter Signalausbreitung sichtbar. Diese Probleme wurden im Detail analysiert und es wurden Empfehlungen gegeben für zukünftige Arbeiten, um den Ansatz produktiv nutzbar zu machen.

Contents

List of Figures

List of Tables

1 Introduction

In recent months with the COVID-19 pandemic situation, governments and society have been looking for easy ways to track contacts between individuals in order to trace chains of infection in a much faster way, than calling people manually by phone. Several countries and technology companies came up with software based solutions implemented on smartphones. One promising approach is the *Exposure Notifications API*, developed in collaboration between Google and Apple, which was then implemented on their mobile operating system platforms [6]. The method that was used in this implementation is based on logging potential contacts using BLE advertisement. The BLE protocol is part of the Bluetooth Core Specification since version 4.0 [13]. This approach of tracing contacts is used by many countries and implemented as mobile applications by the federal health agencies. Examples that can be named are the Netherlands, Germany, Italy, Ireland, South Africa, or several States in the US.

Therefore, devices with an installed contact tracing app based on the *Exposure Notifications API* are constantly sending broadcast messages via BLE, containing a unique cryptographic identifier. If another user of the technology stack is near the sending devices, and if the user receives the broadcast message, it logs the unique identifier of the sender and information about the signal strength. When a person gets infected with COVID-19, the person can upload the received identifiers to a backend server. The backend will publish a list which contains these identifiers. All clients using such an application will compare the identifiers they sent out with this list. When a match was found during this comparison, the person having the match is warned by the application. This warning is sent, because it is likely to have been exposed to an infected person. The person having the match can then take appropriate measurements.

Of course, there is much more complexity and there are country specific configurations implemented, for example in estimating the risk of exposure. This risk of an exposure is assessed by a distance approximation calculated on the signal strength information, which namely are RSSI and TX Power. The distance approximation could be very noisy and error-prone. Studies like the one carried out by Leith and Farrell [46] or the calibration trial for the Singaporean app [36] showed how challenging this approach can be. There are many factors that can

affect the signal strength of a BLE signal. Such factors can be device specific, like antenna design, Bluetooth stack implementation, or sending power. Equally, the factors can be physical. Such factors are always existing when it comes to radio wave propagation: attenuation, shadowing, reflection, scattering, diffraction, and refraction. Due to these limitations, almost no infectious contacts were detected in the above mentioned study by Leith and Farrell. Even if some were detected, the results showed high false-positive rates. Leith and Farrell evaluated the proximity tracing in environments such as a train or a bus.

To avoid such inaccuracies in the measurement of distances, it may be an option taking other signals into account, that have other distribution characteristics. One example could be using 5 GHz signals in addition to the BLE signals for distance estimation. Therefore, the goal of this Master's thesis is to evaluate the use of multi-channel proximity estimation approaches, in order to improve the distance estimation accuracy of privacy preserving contact tracing methods.

In order to improve the existing approaches, we define our focus on IEEE 802.11 signals. This scope is set, because most smartphones do support this technology. Especially, the IEEE 802.11 probe requests frames, which are sent out regularly via broadcast could be utilized. By using these frames, it may not be necessary to implement other transmission methods in existing proximity tracing systems. Consequently, we define the following research questions for this thesis:

1. Is it possible to improve BLE proximity detection by adding IEEE 802.11 signals, either 2.4 or/and 5 GHz, to the calculation and estimation models?
2. Can GAEN BLE advertisements be correlated with IEEE 802.11 probe requests?
    a) How would this affect the privacy of users?
    b) What measures can be taken to protect the privacy of the users?
    c) Can the correlation bring advantages in proximity detection?

The thesis will be structured as follows: In the beginning, we will explain the technical basics that are necessary for understanding this work in chapter 2. We will also discuss related work there. Then, in the chapter 3, we will present our approach to improve the proximity estimation. Furthermore, we will describe our experimental setup, which we will use for evaluation on the one hand, but also to identify problems of the current approach, that is used in Google/Apple Exposure Notification (GAEN), on the other hand. In chapter 4, we will evaluate our approach. Finally, in chapter 5, a conclusion will be given and possibilities to improve our approach in the future will be presented.

2 Background

In this chapter the foundations that are needed for this master thesis will be surveyed. Starting with basic information on the COVID-19 pandemic, which greatly increased the need for digital proximity tracing tools. Followed by fundamentals on contact tracing, for both manual and digital contact tracing. Continuing with a big part on technological details on proximity tracing applications, privacy considerations of such technology and sensor technology used there. Ending with issues that come in with digital proximity tracing systems and presenting another signal type with probe messages that will be evaluated for solving such issues.

2.1 Coronavirus SARS-CoV-2 and the COVID-19 pandemic

In December 2019, clusters of pneumonia cases of unknown source have been detected in Wuhan, China. These rapidly growing clusters raised the attention of Chinese health authorities, who began investigations on the outbreaks then. On January 7 2020, a novel coronavirus (CoV) was isolated by researchers and the fully sequenced genome, named 2019-nCoV, was shared on January 12 with the World Health Organization (WHO) [78]. A test protocol based on real-time polymerase chain reaction (PCR) tests was published shortly thereafter [23]. During the following days and weeks in January, more infections haven been reported in Thailand, Korea, Japan, and more provinces in China. A WHO mission to China then found evidence of human-to-human transmission of the novel virus [77]. In the end of January 2020 the Emergency Committee of the WHO advised the Director-General to declare a Public Health Emergency of International Concern for the novel coronavirus outbreak. By this day $7,818$ cases have already been reported worldwide, with a majority of cases located in China and 82 cases located in 18 other countries. Moreover, 170 deaths were related to virus infections to that point [79]. In February, the International Committee on Taxonomy of Viruses (ICTV) named the virus "severe acute respiratory syndrome coronavirus 2" (SARS-CoV-2) due to its similarity to the corona-virus which was responsible

for the SARS epidemic in 2003 [24]. The disease caused by the virus was named "coronavirus disease 19" (COVID-19) by the WHO. [70] [74]

After several local clusters, the virus hit Europe at the end of February 2020, with clusters in several Italian regions. From there the virus spread over Europe in the following weeks [27]. The situation was declared as a global pandemic by the Director-General of the WHO [80]. At the End of April the number of cases in Europe exceeded the 1 million mark.

The virus continued to spread around the globe. In Europe the number of infections decreased during the summer, but increased massively at the beginning of winter and Europe was hit by a second wave. By December 29 in 2020, the total number of cases worldwide was at $81,451,630$ and $1,778,064$ people died from COVID-19 [26]. To stop the spread of the virus, the WHO continuously issues recommendations for action. An important cornerstone of these measures is contact tracing, to identify and quarantine infectious persons in order to break infection chains [72], which has been implemented and is being carried out by many countries. In the following, the transmission mechanisms of the disease will be briefly discussed and then contact tracing will be introduced.

COVID-19 transmission

The current research state shows that people infected with COVID-19 are most infectious two days before they develop symptoms and early in their illness. Asymptomatic people can also transmit the disease, but it is unclear when they are infectious. The main transmission mechanism is person-to-person transmission with airborne liquid particles. These particles can be larger droplets or small aerosols. The droplets can be emitted by infectious persons for example while speaking, coughing or sneezing. Another person gets infected when these droplets get into their mouth, nose, or eyes. This is likely when people are in close contact. In addition, the droplets can remain on surfaces. Thus, if a person touches the surface and then touches the face, infections are also likely.

Aerosols, which are very small particles that can be suspended in the air, are different. Aerosols can be released by infectious persons through breathing. In certain scenarios infections via aerosols can occur. For example in inadequately ventilated rooms, when the concentration of infectious aerosols in the air is high enough.

Major protection measures against droplet transmission are: keeping enough distance to other persons (at least 1m), wearing masks, frequent hand washing, and covering coughs and sneezes with a bent elbow or tissue. Protection against aerosol infection requires more effort. Only special masks (*e.g., FFP2 or N95*) can filter these small particles. To avoid high aerosol concentrations indoors, frequent airing and ventilation of the rooms should be done. Then, the air mixes with clean outside air and the aerosol concentration is diluted or the room air is even completely exchanged during prolonged ventilation. A far more fundamental measure, however, is to generally avoid contact with others outside one's own household, if possible.

The above presented ways of COVID-19 infection transmissions, represent a summarized state of the current research, that was taken from the *WHO Q&A on COVID-19 transmission* [71]. Despite the measures described, there is always a residual risk of infection, especially if the measures are not adequately implemented. Therefore, it is important to follow up the contact chains of infected persons in order to check whether the contacts have also been infected. When further infected contacts are identified, it is possible to break the following chains of infection in the vicinity of the contact.

2.2 Contact tracing definition

The WHO defines contact tracing as follows: "*Contact tracing is the process of identifying, assessing, and managing people who have been exposed to a disease to prevent onward transmission.*" [75]

The same document defines a contact as a person, who has been in contact with a COVID-19 patient between 2 days and 14 days after disease onset. In addition, the person must meet one of these constraints to be a contact:

- Staying in proximity to an infectious person, within 1 meter for at least 15 minutes.
- Direct contact with an infectious person.
- Caring for an infectious person without appropriate personal protective equipment.
- A setting from a list defined by WHO, such as living in the same household with an infectious person or sharing transportation.

Once contacts have been identified and assessed according to this definition, they must be managed through instructions from a local health authority. Management includes informing the contact and setting orders such as a test or quarantine. The focus will be placed on the identification and assessment of contacts in the following. At first, the manual procedure will be described and then a smartphone based approach which is widely used now, will be presented. The information presented is mostly taken from the WHO interim guidances on contact tracing [75] [76].

2.3 Manual contact tracing

A large number of countries have introduced contact tracing procedures, as proposed by WHO, to break chains of infection. Therefore, health authorities set-up departments who are in charge of contact tracing. The people working there call or visit COVID-19-positive patients and interview them to find out who they have had contact with. This process is the identification phase and normally results in a list of persons who had been in contact with a COVID-19 patient. To obtain a complete list of all the patient's contacts, the agency staff needs specialized social skills. Since the person concerned often does not remember everything or knowingly conceals things in order not to cause distress to other persons. The employees must, therefore, have concrete knowledge to specifically query scenarios in which infections are very likely, in order to recall these situations in the memory of the person to be questioned. In addition, the employees must be emphatic, since the person being questioned is often already frightened enough by his or her own infection. It must be made clear to the person why it is important to identify the contacts so as not to endanger other people. Furthermore, the WHO names community engagement as one of the most important points for contact tracing to work well. Because communities need to understand the benefits of contact tracing, that others can be protected in solidarity, transmissions can be suppressed and participation can avoid stricter measurements. Moreover, public health agencies have to transparently communicate how the information that is shared will be used and how the confidentiality of this information will be guaranteed. The WHO also reinforces that contact tracing and associated steps are not associated with security measures and should not be used outside the realm of public health. When these foundations have been laid and the benefits are clear, people will agree on reporting their symptoms, daily monitoring, naming contacts, and getting tested.

Next, when the contacts of an infectious person were identified, these contacts will also be called by health agency personnel for an interview. In this call, the contact will be assessed, if the person meets the contact definition mentioned before. People who are already positive tested or meet the definition of a contact, will be then quarantined and managed. During this time the contacts should stay at home in self-isolation, and they should have regularly talks to the contact tracing team about their health state. Depending on the strategy followed by a certain country and depending on test capacities, contacts are tested immediately or when they develop symptoms. Therefore, they should directly report the event of developing symptoms to the contact tracing team by themselves. After a 14 day period without developing symptoms, the quarantine is over and a person is allowed to leave the self-isolation.

2.3.1 Resources needed for manual contact tracing

Effective contact tracing can consume a lot of resources. Especially, when the infection rates of a country are high. Many human resources are needed, as well as technical resources. Trained personnel, as previously described, will be needed; reportedly, 30 contact tracers per 100,000 population will be needed as caseloads increase [50]. Besides human resources, technical resources are needed, to make contact tracing more efficient. This includes technical hardware for each contact-tracer and software that supports the tracing process, e.g., in documentation, cluster analysis, surveillance of outbreaks and contact management. An efficient software interconnects local health agencies, to enable tracing across county and municipality borders. A software offered by the WHO, for example, is *Go.Data* [73]. Furthermore, the most important element, the test capacity and the rapid reporting of test results are mandatory requirements.

2.3.2 Issues on manual contact tracing

An article in *Nature* published in December 2020 illustrated several issues in contact tracing [50]:

- Persons who have been tested COVID-19 can not be reached by contact tracing teams. This occurs in England for one of eight people, who have been tested positive. In thin case no further interview and investigation of contacts is possible.

- Positive tested people reached by contact tracers often do not indicate with whom they have had recent contact. Evaluations by the CDC of two counties in the U.S. found that 35% to 48% of those called and reached, report no contact at all.
- When infected persons share information in contacts they had, contact tracers often also have a problem to reach these contacts. In England tracers only reached half of the contacts after three times calling them. Other western countries showed similar worse values. But some countries like Taiwan, Vietnam, or South Korea showed much better values.
- A lack of success monitoring the whole contact tracing process. There is less data reported by the countries doing contact tracing, that enables evaluations of the systems. For example in Europe the ECDC defined metrics and types of data the countries should monitor and report back to the ECDC, but only a few countries is doing so and the data is not readily available.
- Testing and Tracing delay, when one of these factors is delayed contact tracing will get very ineffective. To slow down an outbreak 70% of the cases need to be isolated and 70% of the contacts need to be traced and quarantined.

Most of the named issues result from missing community engagement as the WHO defines it in their guidance [75]. Additionally, another source for such issues like the lack of monitoring or delays, may result from lack of resources or poorly designed processes. Several digital tools can be used in addition to the manual contact tracing, to improve the outcomes. These tools will be described in the following section. Again, it is important to mention that these tools can not substitute the contact tracing, they are just an addition to make things easier and to improve processes [50].

2.4 Digital tools for contact tracing

The WHO classifies digital tools that support contact tracing into three categories [76]:

- **Outbreak response tools**, these are tools that support contact tracing teams in their work processes. As mentioned before, *Go.Data* [73] is such a tool. It helps contact tracers to perform documentation, investigation, analysis of contact chains and generate performance metrics.

- **Proximity tracing / tracking tools** are used to identify contacts of COVID-19 positive tested people. These tools are mostly based on smartphones and the tracking of contacts is done via GPS positions or with the use of Bluetooth signals. There are different frameworks available for proximity tracing, that are based on various concepts, these will be explained in more detail in section 2.8.
- **Symptom tracking tools** are designed to enable a better self reporting of signs and symptoms, that can then easily be reported to the public health agencies. This can make processes more effective, instead of manually questioning people on a daily base about their health state, fixed parametrized values can be submitted by patients to the health agencies. Then, the agencies can perform an automated analysis on the data. The WHO suggests the integration of symptom tracking features into a proximity tracing application.

In the following parts, the focus will be put on proximity tracing applications. Technologies and methods will be explained, available frameworks will be presented, and issues and concerns regarding the usage of this type of applications will be discussed.

2.4.1 Sensor technology

Recent survey papers, such as [4, 53], on proximity tracing applications presented that applications that were put in place to slow the spread of the COVID-19 pandemic mainly use GPS (location based) or Bluetooth as sensor technology for proximity detection. There are also a few hybrid approaches that make use of both technologies.

Location based

The location based approach mainly utilizes Global Positioning System (GPS) or other Global Navigation Satellite System (GNSS) products to track the location of proximity tracing application users. The positions of users are stored and compared with other users' locations in order to identify people, who have been at the same location at the same time, as contacts [76]. GNSS positions retrieved from sensors that are built in smartphones usually have a position accuracy of 2-3m [63]. This gets worse in urban and indoor scenarios, up to no GNSS signal in indoor scenarios. By comparing this accuracy values with the WHO definition of contacts (within 1m distance), one might see, that this may result in high error rates and the positioning is not sensitive enough to detect contacts. But a use

for detecting high crowded areas/situations may be sufficient. However, the use of absolute GNSS positions introduces more issues. When concepts are used, where the location history of a person is transferred to a central service or institution, a persons' privacy may be violated. As the guidance of the WHO describes [75], data and privacy protection is crucial, for communities to use such applications. Therefore, safeguards must be put in place in order to protect this sensitive data and to ensure that it is only used in a public health contact tracing context. These safeguards can be security mechanisms that are either normative legal or technically implemented to secure user privacy. Additionally, GNSS positioning consumes a lot of battery power on the phone, which may result in a smaller base of users.

BLE

This approach utilizes the Low Energy standard that what introduced with the Bluetooth 4.0 specification [13]. With this standard it is possible to send out broadcast messages, so-called advertisements, in short time intervals with little energy usage. There are different frameworks in use, that implement the approach of proximity detection with BLE. The frameworks differ in the content of the messages, the concept of the infrastructure, and other details. But they all follow the same basic principle:

1. *Person A - Alice* sends out advertisements as broadcasts, that contain a cryptographic identifier from *Alice*.

2. *Person B - Bob* receives the advertisements and calculates a risk score, based on the RSSI and the duration signals from *Alice* have been seen.

3. When *Alice* now becomes COVID-19 positive the cryptographic key of *Alice* will be flagged. Depending on the framework a matching will be started, to check who had been in contact with *Alice*

4. Contacts of *Alice*, such as *Bob*, will be notified that they probably had an infectious contact, and they will get further information, for example about self-isolation or about how they will get a COVID-19 test.

Further information and certain details on the various frameworks will be given in section 2.8. In a short outlook it can be stated that there are some solutions available that are very privacy aware, where users have full control on the data, and they decide themselves if pseudonym data about their contacts shall be shared in case of an infection.

2.4.2 Privacy considerations

As mentioned before, the usage of proximity tracing applications and contact tracing in general requires community engagement in order to function well. To foster this engagement and encourage people to voluntarily install a tracing app, good privacy safeguards are needed. Of course, on the other hand, there is still the possibility of making the use of such an app mandatory or de-facto obligatory. For example, by making participation in public life only possible with an app. However, this would reduce the willingness of the population to take other measures to mitigate the pandemic and strong repression would have to be applied to enforce them. In addition, groups without access to technology would be categorically excluded [52]. Therefore, in the following argumentation regarding privacy considerations, it is assumed that the use of such an application will be on voluntary base.

In a comment in Nature Jessica Morley and her colleagues defined a set of guidelines to check if a contact tracing app is ethically justifiable [54]. Therefore, they named four principles that were derived from the European Convention on Human Rights. The principles are: that a contact tracing system must be necessary, proportional, scientifically valid, and time bound. To validate a tracing system against these principles they also gave a check-list with 16 questions. The guide is split into two categories of questions, the first category asks about whether it is right to develop such a system. In the second block it assesses if the system that is developed, developed in the right way. For example if the system is voluntary, questions on data protection, user consent or accessibility. Moreover, on behalf of the time bound argument, the authors claim if a contact tracing is not succeeding, for example in a way that not enough people use it, then at some point the project failed and it has to be stopped. A restart with another strategy is very likely to be unsuccessful, too. Because the engagement already got lost in the first try and it is hard to build it up again.

To provide a better overview of which privacy risks specifically jeopardize the backing for the use of such systems, several risks that play a role in the set-up of such a system will now be discussed below. The scenarios have been presented in the DP3T Framework Proposal [65]. The consortium has made a comprehensive assessment on the subject, as privacy is very much at the centre of their proposal.

Social graph: This graph describes social relationships between individuals. In this graph each individual is represented as a node and if the individuals are in any social relation to each other the nodes will be connected by an edge.

Interaction graph: An interaction graph is build up similar to the social graph, instead of social relations, the edges describe close-range physical interactions between individuals.

Location traceability: This issue addresses location traces. For example, this could be a list of locations an individual has visited.

At-risk individuals: These individuals have been recently exposed to a COVID-19 positive tested person. The privacy related question here is, who can access/control the information of being an at-risk individual.

COVID-19 positive status: Similar to at-risk individuals, the question need to be asked which entities in a system need to know that a certain individual is tested positive. An example for this case is, that users of the system do not need to know that user *Bob from Samplecity* is infectious, for their own protection it would be sufficient to know that they had contact to an infectious individual. They do not need knowledge on personal information of the infectious person *Bob*.

(Highly) Exposed locations This information is similar to location traceability, but in this case the locations visited by an infections individual are meant. And whether the system or certain users need to have this information.

For digital proximity tracing tools, an important issue is the location traceability, the issue described before, has a subset. This is the traceability in a restricted area, for example a mall. The business owners could use the BLE signals, to trace the location of customers, and then use this information, e.g., to influence customers with personalized advertising. An important concept that needs to be mentioned here is the *k-anonymity*. In the case of sending out radio signals, the concept describes, that if the number of devices sending out information, called k, is small enough, for example one. Then it would always be possible to identify the device and there is no anonymity. In many scenarios, it possible to deanonymize users, when the k is small enough [61]. Using heuristics and the information available, it could also be possible to divide devices into groups, to make re-identification easier, as we will describe in the section of IEEE 802.11 probes.

Another tool to evaluate proximity tracing applications on behalf of privacy and ethical concerns was released by the *Chaos Computer Club (CCC)*. Similar to the guidelines of Morley et al. they published *"10 requirements for the evaluation of "Contact Tracing" apps"* [18], based on these requirements a constructive discussion was held in Germany. The government took these concerns serious and changed the plans for an architectural concept from a centralized structure to a more privacy aware decentralized approach [14].

The issues on privacy that the usage of proximity tracing systems bring in, are always an trade-off with the use. For example in order to slow down a pandemic, one might afford the risk of being identified at a location by his or her radio signals. Several frameworks for proximity tracing, that will be presented later in section 2.8, have addressed some of these issues by their architecture. But as mentioned before, due to k-anonymity there is always a residual risk to privacy. In the best case, a proximity tracing system takes all the above mentioned privacy issues into account and encounters them with sufficient safeguards where it is possible. Where it is not possible it needs to be transparently communicated where the users' privacy might be violated and the user need to be asked for consent. Otherwise the community engagement could suffer, and this results in a smaller user base, which makes it harder to efficiently slow down a pandemic.

A prominent case where community engagement suffered, because of a strike against privacy safeguards, was seen in Singapore. There, the government made privacy assurances to their applications users, that data from the proximity tracing system will only be used for public health issues. But the assurances have been reversed and the data was made available to the police, what resulted in public anger and mistrust in the government and the contact tracing application [41]. Even though that people are enforced to use the application, because daily life essentials like access to supermarkets would be impossible, people may get discouraged in the whole contact tracing process. This could result in people being significantly more uncooperative in disclosing their contacts to the health agency in the event of a positive test.

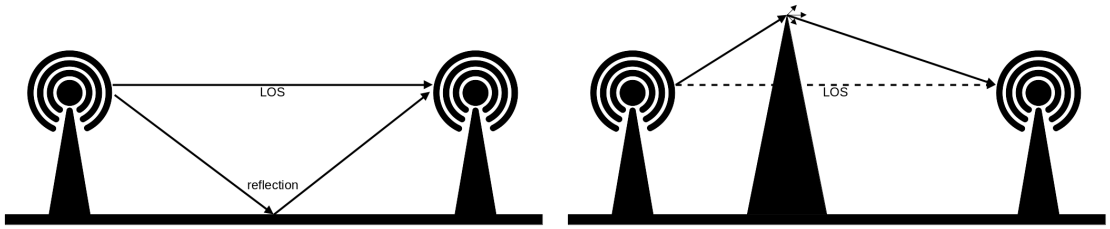
2.5 Radio propagation basics

The transmission of radio signals is affected by several factors, that make it hard to model the propagation of a radio signal. These factors have strong influence on the power with which a signal is received. This measured power is also used in proximity tracing tools to estimate the distance between a sending and receiving device. The energy lost on the transmission path is called *path loss*. The most general issue of path loss is attenuation. When a signal is sent through a medium like air, it loses power in dependency to the distance that the signal needs to travel through a medium. The issues of signal propagation, presented in the following part, shall give some basic understanding that is needed to interpret results of measurement results, that will be presented later, starting in chapter 3. These basics are taken and concluded from the two books [11] and [12].

Reflection When a signal hits an obstacle like the surface of a wall or the ground, depending on the material and the wavelength of the signal, a part of the signal power will be absorbed and the other part will be reflected. The angle of the reflected signal is the same as the angle of incidence and the reflection will cause a phase reversal in the reflected signal. A reflected signal will reach the receiver with less signal power, than a signal received via direct path, the line of sight (LOS). Reflections can also interfere the LOS signals. In figure 2.1a, the issue of ground reflection is exemplary visualized.

Diffraction Signals can be diffracted by touching edges or when propagating into holes. This effect helps at propagating signals to locations that are out of LOS. Diffraction makes it also possible to communicate around the Earth's curvature. As visualized in figure 2.1b, when signal that hits a radio-wave-opaque edge, the signal will be diffracted in the original propagation direction. This phenomenon can be explained by Huygen's principle, which states that each point in the propagation line is a source of secondary emission. Thus, the edge of the triangle in the visualization becomes source of secondary emission, diffracting the signal around the edge.

Scattering If the surface that is hit by a signal is rough, the signal will not be cleanly reflected, as described above. Then, the signal diffuses and scatters in all directions. The amount of scattering depends on the height of the protuberances on a surface, called critical surface height, this is compared to a function that takes the wave length of a signal and the angle of incidence into account. When the function result is greater than the critical surface height, the surface is defined as rough and scattering will occur.



(a) Reflection (b) Diffraction

Figure 2.1: Signal paths resulting from reflection and diffraction [11]
Transmitter Icon by Jejen Juliansyah Nur Agung from the Noun Project

Multipath propagation The previously mentioned issues, e.g., reflections and scattering, can result in having the same signal travelling through the air multiple times. These signals, then arrive at different times at the receiver. The total signal strength of the receiver is a vector sum over the different signals received. Thus, the multiple paths can significantly inflict the signal strength on receiver side. Multipath propagation is an issue that is getting worse in indoor scenarios. Walls, ceilings, the ground, and other obstacles are causing a lot of diffraction, scattering, and reflections, that result in many signals. This makes indoor signal propagation modelling much harder than for outdoor scenarios.

The above mentioned effects can significantly cause interference to radio signals. This, of course, can get even worse, when more senders and clients are transmitting on the same medium. This is often the case, with BLE and wireless LAN, e.g., 802.11b/g, on the 2.4 GHz band. As it will be discussed in the next section.

2.6 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a energy efficient version of Bluetooth which was introduced with version 4.0 of the standard in 2009. It was originally developed for Internet of Things (IoT) use cases, for example to deliver sensor data energy efficient between several sensor nodes. Compared to *classic Bluetooth*, it is possible to send out so called *advertisements* with BLE without having an established connection. These advertisement messages are broadcast messages or beacons that can be received by all BLE clients in signal range. Such as *classic BT*, BLE also operates on the 2.4 GHz band. In the use-case of proximity tracing, advertisement messages are used.

2.6.1 BLE advertisements

For proximity tracing, the advertising physical channel Protocol Data Unit (PDU) is used, within this the PDU type *ADV_NONCONN_IND* is used by most of the proximity tracing frameworks. This PDU type is one of the four legacy advertising PDUs. This type can only be used in combination with the *LE 1M PHY* of Bluetooth and it is not supported for other PHYs, that for example have been introduced in Bluetooth 5. The *ADV_NONCONN_IND* PDU sends out undirected advertisements, which are non-connectable and the devices are not scanable, so other devices cannot request further messages. The packet and header structure of these packets is illustrated in figure 2.2. In most of the frameworks the messages are sent out with randomized advertising addresses (*AdvA*). These random addresses are used when the *TxAdd* field is set to 1. The specific information that are needed by the proximity tracing framework are stored in the *AdvData* field. The particular structure within this field depends on the framework that is used. In version 5.0 of the Bluetooth core specification, *extended advertising PDUs* have been added to the standard. They allow sending increased amounts of data via advertisements. These new PDUs could get interesting for proximity tracing, to send out more information in order to increase the accuracy of proximity estimation. [13]

Link Layer packet for LE 1M PHY										
Preamble	Access Address	Protocol Data Unit (PDU)								CRC
1 byte	4 bytes	2-257 bytes								3 bytes
		Advertising pyhsical channel PDU								
		Header						Payload		
		2 bytes						1-255 bytes		
		PDU Type	RFU	ChSel	TxAdd	RxAdd	Length	PDU Type: ADV_NONCONN_IND (0b0010)		
		4 bits	1 bit	1 bit	1 bit	1 bit	8 bits	AdvA	AdvData	
								6 bytes	0-31 bytes	

Figure 2.2: Structure of BLE advertisement packets [13]

BLE advertisement channels and signal properties

The advertisement PDUs are not sent on all 40 channels that are available for BLE, they are only sent on three specific advertisement channels (*ADV_CHs*). As all BLE channels, these channels are each 2 MHz wide and *Gaussian Frequency Shift Keying* (GFKS) is used for modulation. The three ADV_CHs are channel 37 (2402 MHz), channel 38 (2426 MHz) and channel 39 (2480 MHz). The selection of these three frequencies tried to minimize interference with IEEE 802.11 b/g/n channels 1, 6 and 11. An overview of the 2.4 GHz band with the named channels and also the IEEE 802.11 channels that could interfere is visualized in figure 2.3. But even though that the ADV_CHs have been chosen to have low interference

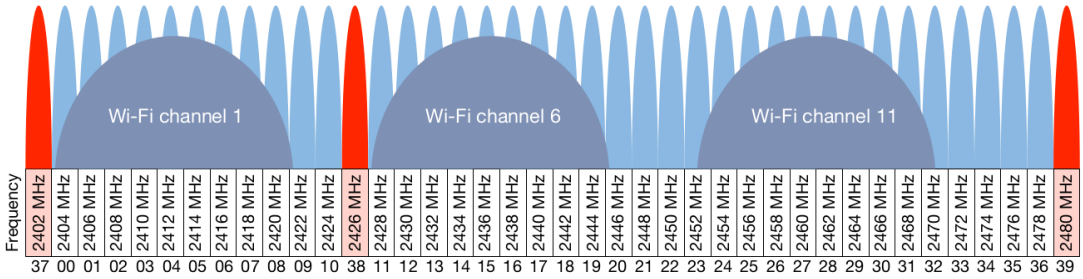


Figure 2.3: Frequencies of the three BLE advertisement channels, shared with data channels and IEEE 802.11 in the 2.4 GHz band. (figure by [56])

rates, studies showed that channel 37 and 38 are still significantly affected by noise and interference from IEEE 802.11. Channel 39 also gets a bit interfered when there are signals with a high channel width are around on IEEE 802.11 channel 11, but the distance between these frequencies is higher than the ones for ADV_CHs 37 and 38. For channel 38 is most likely to be affected by noise, because this frequency gets overlapped by multiple IEEE 802.11 channels (one and six) [56].

Collision and error detection/avoidance mechanisms

BLE utilizes *cyclic redundancy check (CRC)* codes to detect transmission errors or corrupt data on the receiver side. As illustrated in figure 2.2, a 24bit CRC code is sent with each packet. The code is generated prior transmission by the sender, then on receiver side, the packet is checked for bit errors in low layers of the Bluetooth stack. When no errors were detected, the packet is forwarded to higher layers, e.g., the application layer. If an error occurred during the CRC, the packet will be discarded. Since the packets used for proximity tracing are non-connectable, so unidirectional and connection less, no retransmission is possible. This means, that a corrupt packet, is seen as a loss of data, by the application layer [67]. With the new *coded PHY* introduced in Bluetooth 5.0, forward error correction based on redundancy information got available for BLE. But this PHY is designed for long-range scenarios and not used in available proximity tracing solutions [13]. Therefore it will not be further discussed at this point.

In terms collision detection, no explicit mechanisms are implemented in BLE. Except for a random time delay that is added on the advertisement interval for each message that needs to be sent. This delay is used to avoid recurring collisions of two devices sending with the same advertisement interval. However, since only three channels are used for BLE advertisements and the devices do not check whether the medium is free, the probability of collisions will increase with the number of devices and small advertisement intervals. A modelling study for example showed, that with an advertisement interval of 2 seconds, and 300 devices advertising in the area, the probability of packet collisions is at 10% [35]. For crowded facilities, with many IoT devices and proximity tracing clients around, this number can be reached quickly.

2.7 IEEE 802.11 probe requests

IEEE 802.11 probe requests are messages, that are sent out via broadcast, in order to detect which IEEE 802.11 networks are around. Depending on the device, the messages are sent out via 2.4 GHz and/or 5 GHz periodically, when the devices are not connected to a network [3]. Many smartphones send these messages out when they have *screen-on-time*, e.g., when a user is interacting with the phone in order to connect to a network to save cellular data volume. Normally these probe requests are only intended for IEEE 802.11 access points, which then answer with information about the network that is hosted on the access point. But with having a network card in monitor mode, it is also possible to capture these packets on normal client hardware. When the packets are captured, the RSSI can also be captured and information on which specific frequency a packet was sent is available [30].

Compared to BLE advertisements, where it is unclear on receiver side, which ADV_CH was used, this information available for IEEE 802.11 broadcasts is a huge benefit. Because more specific path-loss-models can be computed, such models are in need of the actual frequency to calculate the path loss [34]. Moreover, having this information, it is possible to differentiate traffic that was sent on separate channels. This reduces the variance in the data collected and used for proximity estimation.

2.7.1 Structure

The structure of a IEEE 802.11 probe request frame is illustrated in figure 2.4. The probe request content is stored in the frame body of a generic IEEE 802.11 management frame. In the *frame control* field of the management frame the fame type is specified, which is $0x0004$ for management frame subtype 4. Next, the *duration* field is always set to zero, because the broadcast messages are not acknowledged by other clients, so the access to the medium does not need to be restricted. The *destination address (DA)* field contains the broadcast address $ff:ff:ff:ff:ff:ff$, the *source address (SA)* field has the MAC address of the transmitter, and the *BSS ID* field contains the broadcast address as well. This field is followed by the frame body, which will be described in more detail later. Next, in the *sequence control (seq-ctl)* field, the fragment number and sequence number are stored, where the fragment number is zero in probe requests and the sequence number is counted up, as the number of sent requests. It is cycled at 4096 and reset on MAC address changes. The last field is the *frame check sequence (FCS)*, this is a checksum that is used by the receivers to validate the completeness of the frame. [3]

Generic management frame							
Frame Control	Duration	DA	SA	BSS ID	Seq-ctl	Frame body	FCS
2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	variable length	4 bytes

Probe Request frame				
SSID	Supported Rates	[...]	HT Capabilities	VHT Capabilities
var	var	var	var	var

Figure 2.4: Structure of an IEEE 802.11 probe request frame [3]

When looking into the frame body, there are two fields defined as mandatory by the IEEE 802.11 standard. These are the *SSID* and *supported rates* field. With the 802.11n amendment, the *higher throughput (HT) capabilities* was added as a mandatory field for using high transmission standard [1]. Later, with the 802.11ac amendment the same was done for the *very high throughput (VHT) capabilities* field [2]. Since most modern devices support the *n* and *ac* standard, these devices normally send out these fields in their probe requests.

The length of the SSID field ranges from 0 to 32 bytes. It contains a character string, which is often referred as a *network name*, a human readable name for the actual BSSID. In probe requests a SSID is set, when a client looks for a certain network, e.g., the network the client was recently connected to. If there is an AP hosting the requested SSID, it responds with a probe response. When stations want to retrieve all stations that are around, a probe request with an empty SSID field is send. This zero-byte case is called *broadcast SSID* [31].

Next, we will have a look on the supported rates field, it consists of a string of bytes, each byte describes a transmission rate that is supported by the sending device. The Rates field can also be declared as mandatory, therefore the seven low order bits of the byte are used to encode the rate and the most significant bit (MSB) is used as flag, to indicate whether the rate is mandatory. Up to eight transmission rates can be encoded in this field. If more rates are supported, the field *extended support rates* can be used to encode these rates [31].

For 802.11n clients, the probe requests also contain an HT capabilities element. It has a size of 28 bytes and contains the following parts [32]:

Name	**Size [byte]**	**Description**
ElementID	1	Set to 45 to indicate that this is a HT capabilities element
Length	1	Defines the length of the element, for this ElementID it is 26
HT Capabilities Info	2	Contains several options encoded in bits, such as supported channel width, supported encoding, etc.
A-MPDU Parameters	1	Declares A-MPDU parameters, the exponent length and minimum spacing
Supported MCS Set	16	Defines the supported modulation and coding scheme set (contains supported data rates, etc.)
HT Extended Capabilities	2	More space for supported protocols, like phased coexistence operation
Transmit Beamforming	4	Information about supported beam forming methods and modi
Antenna selection	1	Used by device that have more than one antenna

Table 2.1: Fields of an HT capabilities element of a IEEE 802.11 probe request

Similar to the 802.11n capable devices, there are many devices that do support 802.11ac, these devices send VHT capabilities, containing the following information [33]:

Name	**Size [byte]**	**Description**
ElementID	1	Set to 191 to indicate that this is an HT capabilities element
Length	1	Defines the length of the element, for this ElementID it is 12
VHT Capabilities Info	4	Contains information about the protocol features and transmission rates that the transmitter supports
Supported VHT MCS & Number of Spatial Streams Set	8	Describes the receiving, and transmission capabilities (4 byte each), and also contains the highest total data rates supported in 1Mbps units

Table 2.2: Fields of a VHT capabilities element of an IEEE 802.11 probe request

The IEEE 802.11 standard defines more elements that can be included in probe requests, for example, more specific information via a query in the *request* field or vendor specifics. However, the standard case for newer devices includes the above mentioned and more in detail explained mandatory fields [37].

2.7.2 Collision and error detection/avoidance mechanisms

On IEEE 802.11 several mechanisms are used to ensure a packet is received without any errors. There are several points of failure that may occur. At first, it needs to be ensured that the channel where the message should be sent on is free. Otherwise, collisions may occur. To avoid this, *Carrier Sense Multiple Access Collision Avoidance (CSMA/CA)* is used. In this procedure a client first listens on the channel, to check if another client is sending at the moment. If the channel is occupied by another client, the client that wants to send is waiting for a random amount of time and then checks again if the channel is free to send. When the channel is free, the message can be sent. Since there is still a little risk for a collision on the receivers side, it can not be completely assured that the packet is properly received. To encounter this problem, CSMA/CA utilizes acknowledgement packets, that are sent by the receiver on a successful transmission. When a sender does not receive an acknowledgement packet in a certain amount of time, the original packet will be retransmitted. For probe requests, the listen before sending procedure is also used. But since packets are sent to broadcast, and it is unclear if there are any clients listening in the area, no acknowledgement packets are used for broadcast probe requests [3].

Next, when the packet was received successful, it needs to be checked if the packet was received completely or if it is corrupt. Therefore, a checksum, as already mentioned in the management frame format, is sent with each IEEE 802.11 packet. On receiver side a checksum of the frame will be calculated and compared to the checksum that was included by the sender. When these checksums match, the packet was received completely, otherwise a retransmission will be requested. There are also *forward error correction (FEC)* mechanisms in IEEE 802.11 available, that encode redundancy information over a set of frames. When single packets get lost, it is possible to rebuild the missing frame on base of the redundancy information of the other frames. This can reduce the number of retransmissions. However, on probe request frames FEC is not used, since only single broadcast frames are sent [3].

2.7.3 Privacy issues with probe requests

Probe requests can be used to identify devices in a certain area. Several older devices do not use rotating random MAC addresses to send out probe requests. With this information, an attacker can determine that a device or person entered a certain area, running more sniffing infrastructure it would be also possible to determine the location of a device. This attack scenario got more difficult, with the introduction of MAC address randomization. For example, for Apple smartphones this feature is supported since the *iPhone 5*[1]. The fact, that this function is activated on many modern devices has made the attack more difficult, but not impossible. Several studies have shown that there are efficient methods that allow fingerprinting of IEEE 802.11 stations to identify the devices again [30] [68]. One promising and accurate working approach, for example, is using the HT and VHT capability information of a station to do a fingerprinting [37]. These fingerprint approaches are also sometimes used to count the number of people in a certain area. To avoid a identification of recurring stations, and just to count people, for example, recently a more privacy preserving approach using the timing of probe requests was published [64].

Next, an overview on the commonly used frameworks for proximity tracing to encounter the COVID-19 pandemic and more technical details on the proximity tracing method using BLE will be given.

2.8 Contact tracing frameworks

A general overview on the workflow of proximity tracing by using BLE was already given in section 2.4.1. A selected number of frameworks will be presented now. The choice consists of the two most discussed approaches in Europe, Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) and Decentralized Privacy-Preserving Proximity Tracing (DP-3T), and GAEN, the framework implemented by two big smartphone operating system manufacturers, which was developed on the base of DP-3T. There are several other approaches, that put less effort on privacy and data protection, but since this topic was very important to most of the EU countries the discussion was only between these two approaches. Except for a few countries like Cyprus, Bulgaria, Hungary, or the Czech Republic who developed own proprietary solutions. However, these approaches mostly didn't succeed with about or less than $100,000$ installations. A comprehensive overview and discussion on more approaches is given in two survey papers [4, 53].

[1] `https://support.apple.com/en-gb/guide/security/secb9cb3140c/web`

2.8.1 Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT)

This proximity tracing protocol was developed by a big consortium of scientific institutions, companies and public health institutions. The approach was promising and in first place followed by the German and French Government. The infrastructure consisted out of a central backend that should be run by health authorities and a smartphone application that should be installed on the smartphones by people participating in proximity tracing. There are two known PEPP-PT implementations. One is *ROBERT* [17] from France and *PEPP-PT NTK* [59], which was a proof of concept implementation in Germany.

When a user installs the application, it queries the backend for registration. To avoid masses of registration requests and DoS attacks, the client has to solve a proof of work (PoW) challenge, in this case an encryption task, and then the user has to solve a captcha. When this was successful, the application on the client side will receive OAuth2 credentials from the backend. Additionally, the backend generates a unique 128-bit pseudonymous persistent user ID (PUID) and a push notification ID. The notification ID is then used to send notifications to a user [53].

With the PUID and a secret key for a time interval a set of ephemeral BLE IDs (EBIDs) will be generated using AES encryption for each user by the backend. These IDs will then be sent to the application on user side and broadcasted in the BLE advertisements. Other users in the signal range then receive such broadcast messages, their applications log the EBID and the signal strength (RSSI, TX power and other metadata) locally. The log is stored for an epidemiologic sufficient time (presumably 21 days) and then deleted [53].

If a user of this system now gets infected and tested positive, a health authority will provide a token, that the user is legitimated to upload his or her contact log to the backed. On backend side then an algorithm is doing a risk estimation for each EBID that is in the contact log. When an encounter is classified with a high risk, the user with the derived PUID from the EBID will get notified. Depending on the implementation this is done in two different ways. In ROBERT the clients regularly poll the backend, asking for their risk state. And in PEPP-PT NTK, the client application will get a push notification, that their state changed and that a polling should be performed. The push notifications in this case are mixed with other random PUIDs, to make the data more noisy and reduce the risk of identifying interactions graphs and at-risk individuals. [17] [59]

This central instance of knowledge (the backend) was a big controversy in the consortium. Many members quit and 300 researchers reached out to the governments, arguing for a decentralized approach in an open letter [42]. Members who quit the consortium then formed the DP-3T initiative with several other players, who then offered a decentralized approach. As previously mentioned in the privacy concerns section, the German government switched away from PEPP-PT to a decentralized approach and in Europe only France is running a system on basis of the PEPP-PT protocol [53].

2.8.2 Decentralized Privacy-Preserving Proximity Tracing (DP-3T)

The DP3-T protocol is following a decentralized approach, instead of a centralized one like PEPP-PT does. Decentralized in this manner, means that there is no single central instance/component where knowledge and functionality are aggregated. Thus, the risk checks and comparisons of contact logs or the generation of temporary IDs shouldn't done on a central server. To overcome this issue, former members of the PEPP-PT consortium proposed a new protocol called DP-3T. The new consortium draw three protocol designs, a low-cost, unlinkable and a hybrid design. The main difference in these designs is the generation of temporary IDs and which data will be sent to the backend. The information presented in the following part are concluded from the DP3-T white paper [65] and partially from the survey paper by Martin et al. [53].

But to start from the base, for DP3-T a backend service and an application on the smartphones of the users who want to participate is needed. When the application is installed on a smartphone a number of ephemeral IDs (EphIDs) will be generated. Depending on the design these IDs are generated in different ways. For the low-cost and unlinkable design the EphIDs are generated using a secret key. The key is changed every day by applying a hash-function onto the key of the previous day. The secret key is then used to generate the EphIDs with a pseudorandom function and a pseudorandom generator. In the hybrid design, random seeds (changing every day) are used to generate the EphIDs.

These EphIDs are then broadcasted in BLE advertisements. Each EphID has validity for a few minutes. Then, the next EphID is taken to avoid tracking. If another user receives such a broadcast message, the EphID will be stored locally on his or her phone and keeps the contact log for about 2 weeks.

When a user is tested positive, he or she will receive a token from the health authorities, with this token it is possible to upload the secret keys or the random seeds to the backend. Only the keys or seeds are uploaded that lay within the time period of infection. Other clients participating in the system periodically poll the backend for these keys/seeds and download them. Then, each participant calculates EphIDs from the downloaded key/seed for the specified time interval and compares it with the EphIDs that are stored in their contact log. If a match occurs, a risk calculation on the local device will be done. This calculation is based on the received signal strength, the actual implementation, and thresholds shall be defined by health authorities. Switzerland for example is doing a risk assessment per day, so a risk value is calculated over all encounters of a day.

The consortium of DP-3T carefully discussed privacy and security issues in their white paper. Moreover, there was a big public discussion on this, after the CCC and other NGOs heavily criticized the centralized PEPP-PT approach, they have demanded to use a decentralized approach, which was namely DP3T at this time [19]. Meanwhile, when this public discussion was on, Google and Apple decided, to implement a proximity tracing framework on their mobile operating systems iOS and Android. Their framework offers an Application Programming Interface (API) for countries. They can develop a proximty tracing app using the advertising and tracing features that are directly implemented into the operating system. Many countries did so, since it was easier and faster to bring up a proximity tracing solution. Google and Apple took the DP-3T hybrid design as their main inspiration for the development of their framework [25]. Next, details of the GAEN framework will be presented.

2.8.3 Google/Apple Exposure Notification (GAEN)

The GAEN framework is used by at least 15 EU member countries and several other countries and states around the world to implement their proximity tracing solution [53]. Germany, for example, implemented their *Corona Warn App* (CWA) using GAEN. This application has been downloaded more than 25 million times, from the *Apple App Store* and *Google Play Store* [60]. As described before, the foundation of this framework is the DP-3T protocol. Therefore, the basics will not be explained again in this section. In the following, some more technical details will be given. Starting with the messages actually sent by participating devices, the key generation and management, and details on the distance estimation.

A BLE advertisement sent out by GAEN does carry its' information in the payload field (AdvData). Starting with the $0x01$ flag set to true, this stands for the BLE general discoverable mode and basically informs receiving devices that this is a general broadcast advertisement. Next the *Service UUID* follows, this field is set to $0xFD6F$, the service UUID of the exposure notification service. It is used for identifying the purpose of this packet. Same holds for the *Service Data UUID*, it identifies the service data, using this UUID clients know how to interpret the following bytes. These 16 bytes are the Rolling Proximity Indentifier (RPI) which is comparable to the EphID, followed by 4 bytes of associated encrypted metadata. The metadata contains information of the framework version that is used and the transmit power (TX_power) level, describing the power that is used for sending out the advertisement. It is later used to improve the distance approximation. The overall advertisement payload is visualized in figure 2.5.

Flags			Complete 16-bit Service UUID			Service Data - 16 bit UUID				
Length	Type	Flags	Length	Type	Service UUID	Length	Type	Service Data		
0x02	0x01 (Flag)	0x1A	0x03	0x03 (complete 16-bit Service UUID)	0xFD6F (Exposure Notification Service)	0x17	0x16 (Service Data – 16 bit UUID)	0xFD6F (Exposure Notification Service)	16 bytes Rolling Proximity Identifier	4 bytes Associated Encrypted Metadata

Figure 2.5: Structure of the advertisement payload sent out by GAEN [8]

Key management The RPIs that are sent out with the BLE advertisements are generated in a rather complex manner. It all starts with an Temporary Exposure Key (TEK), that is generated periodically locally on the device, every 24 hours, using a cryptographic random number generator. Using this TEK and a discrete representation of the current time as input for an SHA-256 function, an RPI key is generated. This RPI key has a validity for one day, just like the TEK. From the RPI key and a padded data squence the actual Rolling Proximity Indentifier (RPI) is derived using AES 128 bit. The RPI always changes at the same time as the Bluetooth MAC address. Same holds for the encrypted metadata. It is encrypted using AES in counter mode with the current RPI and a metadata key that is also derived from the TEK.

When participants are diagnosed positive, for example, in the German CWA, they have to prove the validity of the test by scanning a QR code or calling a hotline to obtain a TAN. If the test is valid and positive, the user can upload his or her TEKs and according timestamps that describe the validity of the TEKs. Only the keys of a specific time window, e.g. the time when the user could have been infectious, will be uploaded. After that, the procedure continues as described in DP-3T, the backend aggregates the TEKs of all positive tested users and makes them available for download to other participants. These participants then regularly poll the backend and download the published keys. Next, each other participant derives the sequence of RPIs from the TEKs and compares it against the RPIs that have been logged from BLE advertisements. When a match is found, the metadata gets decrypted and a risk assessment will be done by estimating the distance to the infectious individual and the duration of a possible exposure. [9]

Distance estimation Once a match is found and the metadata is decrypted the attenuation will be calculated, to estimate the distance between the two individuals, using this formula:

$$\text{Attenuation} = \text{TX\_power} - (\text{RSSI\_measured} + \text{RSSI\_correction})$$

Where TX_power is obtained from the metadata, RSSI_measured from the local log and RSSI_correction from a list where the vendors submitted correction values for various smartphone models. Since version 1.5 of the API and on smartphones with Android 9 or higher, for each scan window the minimum and average attenuation values are available. Due to the fact that the three ADV_CHs may have different noise levels, that result in various RSSI, it is recommended to use the minimum attenuation for further risk assessment. It describes the best signal transmission situation over the three channels. In older API version only the average was used over three channels and this may have resulted in higher attenuation values and then resulted into false negatives. In older Android versions only channel 37 was used [7]. The next step could vary depending on the implementation of the actual proximity tracing app, different thresholds for calculations are used. The following thresholds are from the German CWA, for example. Basically the framework is calculating a weighted risk score. By first classifying the attenuation in *very close*, for distances less than 1.5m, with an attenuation value less than 55dB, *close*, for distances less than 3m, with an attenuation value less than 63dB or safe distances for all higher values. These classified attenuation durations can also be obtained by querying the API.

Next, the time is added up for the two close cases. These sums then go into the following formula:

$ES = B1 + 0.5B2$

With $B1$ as the time of exposure that has been less than 1.5m and $B2$ being the time of exposure less than 3m. If the result is at least 15, the user will be warned by the application and informed about further proceeding *(e.g., getting a test etc.)* [62]. The attenuation thresholds vary, depending on the country specific implementation by the health authorities.

In the later parts of this thesis GAEN and the previously described distance estimation procedures will be taken for comparisons, in order to improve this procedure. Due to the issues with this method, that will be described next.

2.9 Issues and problems of digital proximity tracing

Several issues are connected to the presented method of digital proximity tracing.

1. No country in Europe or America, which runs voluntary based proximity tracing systems, has broken through the user mark of 60% of the population. These 60% are defined as a threshold that is needed to bring the outbreak under control by using digital proximity tracing [29]. But this does not mean, that rolling out and using such an app, with a smaller user base is ineffective. Also, user numbers apart from that 60% can help to identify possibly infectious contacts, help to quickly alarm people at risk, and help to reduce the work-load of the contact tracing teams. The higher the user base, the higher the likelihood that contacts with COVID-19 positive persons will be quickly traced [57].
2. The proximity estimation using BLE is prone to errors and comes with high false negative ratios in various environments. A calibration study and studies in real life scenarios such as a bus [48] or a tram [47], showed that it is really challenging to perform an accurate proximity estimation based on the RSSI values retrieved via BLE. It is not even necessary that the RSSI value decreases with increasing distance and this will make it even more challenging to get an accurate distance approximation [45]. Even more studies from the field of indoor navigation attest that distance measurements using BLE are difficult due to multi-path effects [21] [15] [28].

Moreover, as mentioned in the signal properties part of BLE advertisements, interference by IEEE 802.11 on 2.4 GHz can influence the RSSI values as well and in indoor scenarios multi-path effects are making the measurements even worse [56]. For example, scenarios such as the tram mentioned above, triggered no alerts using the Swiss or German rules for estimating the risk of infections [47]. Updates of the GAEN framework already tried to face this issue by using the minimum attenuation of a scan window. However, it is hard to overcome physical issues of BLE signal distribution with software updates. Google and Apple also state by their own that attenuation is a very noisy distance proxy and that it is necessary to make a trade-off between precision and recall by defining the thresholds for risk estimation [7]. The studies that were briefly mentioned here will be discussed in more detail in the related work section.

3. Digital proximity tracing solutions based on smartphones are not accessible to people without a smartphone. These are not only elderly people. This sometimes might also come with the constraint that people do not want to use such a technology. This will make it even more difficult to reach a number of users higher than 60% of the population. A government that is running a proximity tracing system needs to take this into account and has to develop a strategy that also protects this group of people, in a way that a digital proximity tracing system is the part of an overall strategy to fight a pandemic [76].

Two of the here three named issues are from a less technical area and require action by governments and the society, for example to increase the encouragement, that more people are using such a proximity tracing application. But the problem of inaccurate proximity estimations is a technical problem, and finding methods to improve the proximity estimation will be the focus of the following parts in this thesis.

2.10 Machine Learning

Some techniques from the field of machine learning will be used in this thesis. The explanation why a certain method is selected will be given at the place of usage. Next, the algorithms decision tree, and random forest will be briefly described. The algorithms presented are simplified and summarized from the book “Introduction to machine learning” written by Ethem Alpaydin [5].

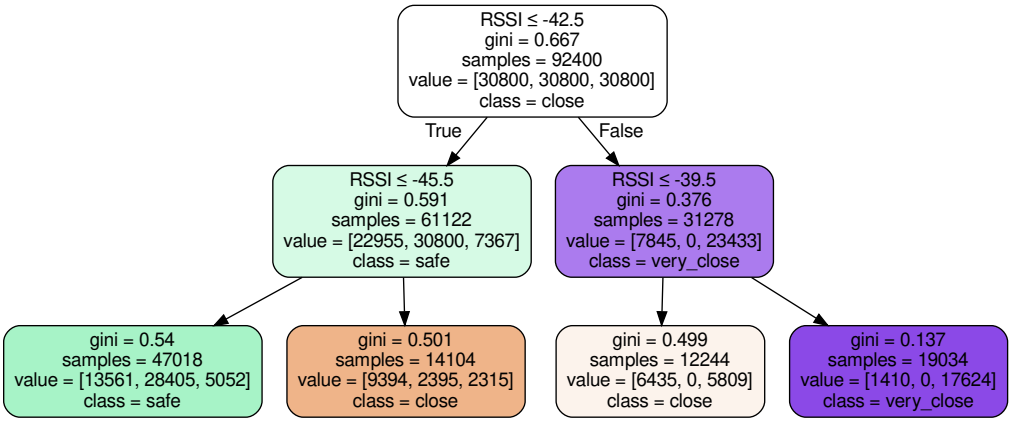


Figure 2.6: Decision tree example for splitting into three discrete classes on base of a RSSI value

2.10.1 Decision tree

A decision tree is a hierarchical data structure that is used to solve a classification or regression problem. A classification tree is used to assign discrete classes to observations and a regression tree can be used for continuous values. Each node of such a tree, defines a split of an observation set. The split is performed by applying a threshold to a given feature. This can be for example, a split for all observation having an RSSI ≤ -42.5. By performing these split in various layers of a tree, the number of observations that will end up in the leafs will be reduced, the samples that end in the leafs also represent the classification result. An example of a decision tree, that classifies observations by RSSI into three discrete distance classes, is visualized in figure 2.6.

The splits of a classification tree are estimated from a training data set. An impurity metric is used to estimate the quality of a split. In the case of only one feature, as shown in figure 2.6, the variation of the feature can be iterated, and when the split with the highest impurity value is found, the root split is found. The next nodes are build up the same way, either until no more splits are needed, which is the case when the resulting sets are completely pure. This means that there are only samples of one class in a set. Another convergence criterion is when the tree reached a predefined depth-level, as in the presented tree. Popular metrics to measure the impurity in building a decision tree are Entropy or the *Gini Index*. Another method to measure the quality of splits, is to use the misclassification error.

Random forest

Several classification trees can be combined into a forest. A special case of this combination is the random forest method. From the complete training data-set, random samples are taken, with these samples several trees are trained. The samples are taken with replacement, this means a sample can occur multiple times in a set. This concept of training is called *bagging*. After training the trees, all trees have to predict a class on a new unseen sample. Then the average result from all predictions is taken as the final result. There are also weighted votes possible between the trees. By using this ensemble method, of predicting with multiple classifiers trained on smaller samples, the models usually have low correlation to each other. By combining the result, the individual error of the over-fitted trees fell away. Such forest usually produce more robust classifiers and better classification results.

2.11 Related work

In this section we will present and discuss studies that investigated and evaluated methods of proximity tracing using BLE signals. Moreover, studies on improving existing proximity tracing methods will be also discussed. At the end of this section, we will additionally present papers that analysed the state of users' privacy in applications that use the GAEN framework.

2.11.1 Evaluation studies on proximity tracing using BLE

The following part will of this section will focus on studies, that evaluate proximity tracing methods and systems using BLE.

GAEN evaluation studies by Leith and Farrell

Douglas Leith and Stephen Farrell brought out several papers, when the deployment of proximity tracing applications started. They investigated in different scenarios the accuracy of distance measurements using BLE. In their first studies, in which they investigated on the accuracy in a bus and in a tram, they used a modified version of the exemplary implemented proximity tracing app using GAEN. This app and an advertiser app were installed on several handsets. With this they collected data in various scenarios in a train and a bus. From the data collected, the attenuation was calculated. Then, the duration summarized for which a device was seen with a certain attenuation.

Next, these durations were compared to the thresholds of different proximity tracing applications based on GAEN to estimate the proximity. In these studies they concluded, that almost no detections of close distance were triggered and that the performance of such systems comparable to just randomly triggering proximity notifications [47, 48]. The findings presented here, need to be interpreted with some remarks in mind. First, the evaluation was done using an GAEN API that was in quite early state. In this version the observations from all three ADV_CHs are used and an average is taken per scan window. Due to the fact that the RSSI between that three channels can have high variance, the attenuation can get worse by averaging, than it was actually measured on a less interfered channel, which results in poor classification results. This issue was fixed in later GAEN versions, by only using the minimum attenuation, that was observed in a scan window. Thus, only the channel with the best performance is used for further calculations. The issue of using the average of the three ADV_CHs will be also investigated and presented later in the experiment results of this thesis. Second, proximity tracing systems, e.g., the CWA that was deployed in Germany, perform a post processing on the measured data. In this post processing the risk factor of proximity is multiplied with a transmission risk factor. This will result in warnings that will also be triggered for smaller risk duration values. However, the impressing key insight of these studies is, that there is no clear trend between attenuation or RSSI values and distance.

In a study later in 2020 [45] they put the focus more on this issue of low correlation between distance and RSSI. They compared several scenarios, outdoor and indoor scenarios, including the tram scenario. In this study they showed that a correlation between RSSI and distance can be observed in outdoor scenarios. But when it comes to indoor scenarios this correlation gets worse. Moreover, they presented that factors like the orientation of the device and the location, so whether it is in a pocket or outside, and if there are human bodies in the LOS can have crucial influence on the RSSI values that are observed. In this work, they concluded that it is likely to be difficult and time-consuming to develop an accurate method for proximity detection using BLE.

Proximity estimation accuracy survey of Bluetooth-based contact tracing apps

Zhao et al. evaluated a set of proximity tracing applications, surveyed their broadcasting behaviour, the underlying calculations for proximity estimation, and tuning possibilities to the RSSI measurements. They showed that there are different internal factors in software and in hardware, which can affect the RSSI. For example chipset and antenna of the smartphone, or the operating system running

on a smartphone that is adjusting the levels of transmission power. Moreover, they briefly mentioned issues of signal propagation like obstacles and interference of other signals. In conclusion, they are suggesting tuning the RSSI calculation by devices specific coefficients and taking the TX_power used by the sender into account. Additionally, they criticized, that only a few applications, from the set they have surveyed, are performing an RSSI parameter tuning [81]. However, the proposals they made are already included by GAEN. The packets sent out contain the TX_power in the meta-data field, and a device specific calibration is done. Therefore, Google built up a long list of devices and calibration factors in cooperation with vendors [7, 8].

BLE based proxmity estimation used for indoor navigation

Since BLE is part of the Bluetooth standard, researches from the field of indoor positioning and navigation, tried to utilize this technology to achieve better positioning results. Some studies here discussed the proximity estimation between two devices and others showed impressing surveys on the signal propagation properties of BLE. Some of those studies shall be briefly discussed here:

Chowdhury et al. discussed the proximity estimation via BLE between two devices. They showed, that there is no clear trend between distance and RSSI observable in indoor scenarios. The model they presented, utilizes smoothing the RSSI over a given time-series. Moreover, for certain RSSI ranges they then applied three different models to estimate the distance. Their model achieved an overall Root mean squared error (RMSE) of 13.4% [21]. However, it is hard to apply this approach to a proximity tracing scenario, like we have in the COVID-19 pandemic. This is because there is no large time-series available to perform the preprocessing averaging smoothing steps on. Moreover, a limiting factor of the presented study is, that the model was only tested in one environment and only with one device. It is likely that other environments which are more reflective, or other devices with different RSSI properties will cause much higher error rates.

Čabarkapa, Grujić, and Pavlović published a survey paper on the various positioning techniques that are using BLE. For beaconing and path loss approaches, which can be used between to devices, they state a probability of 50% accuracy to estimate that another device is in proximity of $1.5m$. This result was achieved with an adaptive path loss model and a particle filter applied to the data. Overall, they concluded if accuracies higher than 80% indoors in distances less than $1.5m$ are needed, a calibrated infrastructure of reference points that captures the advertisement signals is needed [15]. Meanwhile, after some more years of research, the accuracy of beacons and lists of calibration factors for various devices of course improved. But still, this study draws how challenging proximity estimation indoors on short distances can be.

A study done by Faragher and Harle, that in the end evaluated a fingerprinting solution, brought a very good overview in signal propagation issues with BLE advertisements. They showed that a human body can cause about $10dB$ of attenuation to an advertisement signal, which then can have heavy impact on the range estimate. Moreover, they described the difficulty, of not knowing the actually used ADV_CH in advertisements. Without knowing it, it is hard to overcome and model fast fading effects that occurs from interference and the small bandwidth used in BLE. These multipath fading effects could cause drops of $30dB$ in some cases. They used modified software on a iPhone to determine the channel and removed the multipath effects applying batch smoothing. Their tests showed that a batch size of 10 can achieve good results. Another interesting insight from their study is, that IEEE 802.11 signals in 2.4 GHz band seem to be less susceptible to these fast fading effects. They showed that even in static environments and with device specific models it is challenging to handle this issues. After the signals were filtered and smoothed they performed a fingerprinting approach, for certain locations to estimate the position [28]. Since they applied their filtering on constantly moving devices, their approach in smoothing deep fades out is difficult to use, even more with not knowing the channel. In proximity tracing systems deployed in large scale it is not possible to do such changes to the users devices. But they also suggested only taking the best signal that was received in a scan window to overcome this issue a bit.

Chen et al. presented their *LocBLE* approach in 2017 to perform indoor positioning. They used a rather complex pipeline, to mitigate the issues of BLE at stake. Their approach included a detection for environmental changes, to estimate whether there is a LOS, a partial LOS, or no LOS. To detect this environmental types they used a support vector machine (SVM) with a linear kernel. The raw RSS value that should be used for proximity estimation, is preprocessed with a Butterworth filter to handle fast fading issues and the result is then computed in an adaptive Kalman filter, to remove the delay, that was introduced by the Butterworth filter. Then, they used a regression approach to estimate the distance. Their approach was extended to also handle moving targets by utilizing the motion sensor of a phone, and they also proposed a calibration method, using a clustering approach between multiple beacon signals. Overall, even though they had a rather complex preprocessing pipeline and having multiple signals from different beacons, in different scenarios, their resulting positions had errors, which were ranging from $1m$ to $2.4m$. They justified these errors with the present multipath effects indoors with BLE [20].

There are several other studies from the field of indoor positioning that have been found, most of them draw similar insights that the few sets mentioned before. There are also studies that only utilize IEEE 802.11 signals or studies that compared IEEE 802.11 to BLE positioning, like [69]. This study also showed that IEEE 802.11 in 2.4GHz produced rather good results, and BLE was quite error prone, but the accuracy got also a bit better when they used smoothing filters. As in some other studies they used an infrastructure with many reference points. The advocated for BLE solutions since it is easy to deploy and cheaper than an IEEE 802.11 positioning infrastructure. Additionally, it can be said, that there are only a few studies that evaluated hybrid approaches with both technologies, so BLE and IEEE 802.11. One of these few studies is from Terán et al. [69]. They combined RSS values from IEEE 802.11 and BLE in a machine learning approach. It was not stated which IEEE 802.11 band they used. With combining this two signals in a k-NN classifier, they achieved an accuracy of 75% with a resolution of $1m$. Which was slightly better than just using IEEE 802.11. Again a limiting factor of this study is that it was only carried out in single environment.

Detecting social interactions using Bluetooth RSSI and machine learning

Palaghias et al. tried to assess social interactions via Bluetooth RSSI values. They presented a two-layer machine learning approach, with some specialized models for each of the three distance classes in layer one, on second layer then the membership of the actual class was mapped. They claimed that the approach they presented has a accuracy up to 93.52% in distance estimation. Moreover, they successfully compared their approach against a state-of-the-art RFID approach [58]. It is not directly stated, but it looks like, their measurements used for training and evaluation were all captured in an office environment. Furthermore, in all experiments the same device type was used for sending and capturing. In our opinion it is likely, that this high accuracy values, may result from over fitting to this specific scenario. Besides that, the study does not state which Bluetooth signals were used for evaluation, so it is unclear whether they used BLE advertisements or if they used classic Bluetooth connections, or something else.

Proxmity tracing and indoor challenges using classic Bluetooth

Liu, Jiang and Striegel also did a study on assessing the accuracy of Bluetooth in proximity estimation. Their study was carried out in 2012, at that time BLE was not available, so it was carried out with classic Bluetooth. They did measurements in several scenarios, but with only one device type. And also showed, the difficulty of indoor scenarios. They applied a single and multiple threshold approach to their data, to estimate the distance. The most interesting factor from this study is, that they used the light sensor of a phone, to estimate the environment, so if the user is indoors or outdoors. Based on the environments they applied the thresholds. Their focus was more on showing that Bluetooth is a sufficient way for proximity estimation, challenging IEEE 802.11 and GPS approaches. Therefore, they tolerated an error of $1m$ to $1.5m$ [51].

Emprical study on signal propagation properties for the BLE advertisement channels

Regarding the specifics in signal propagation of the three ADV_CHs, Nikoukar et al. did a very informative empirical study. They collected data in various indoor and outdoor scenarios. Additionally, they characterized the noise floor, for each channel in each environment they measured and proposed values that can be used in modelling signal propagation for these channels. As already pointed out in the presentation of other studies, a trend of RSS against distance can be obtained from outdoor scenarios, but it is quite hard for indoor applications. In both situations they showed, that the three channels can have high variances in RSS to each other in the same distance. This variances get even higher indoors. They explained this reflections, multipath effects, interference by IEEE 802.11 networks inside, and antenna anisotropy. Moreover, they state that from these three channels, channel 39 is facing less interference and the measured values have less variance due to smaller IEEE 802.11 interference [56].

2.11.2 Improvement approaches of proximity tracing

In the following part, we are going to present recent studies, that had the goal to improve proximity tracing systems, which were used in fighting the COVID-19 pandemic.

Improving BLE proximity estimation using network localization algorithms

Clark et al. suggested an approach of applying network localization algorithms to the collected BLE RSSI values, of the various participants in a proximity tracing system. Their results showed that in less noisy quite optimal environments, a direct estimation using the RSSI values can achieve the best results. But in environments with more noisy and not having RSSI data between all participants available, network localization algorithms, such as *SDP* can achieve good results and can likely identify all true positives in a distance less than $2m$ [22]. But the usage of such algorithms is much more complex, than a partitioning based on thresholds. Moreover, to compute such algorithms, a central instance needs to collect all RSSI values and timestamps captured by each participant. This data is needed in order to build up the network graph. This can highly violate the privacy of the participants of such a system, the central entity or an entity that is able to compromise the data from the central server, would be able to build up an interaction graph for all users.

Improving BLE proximity estimation using machine learning approaches

The U.S. National Institute of Standards and Technology (NIST) called AI researchers for a contest in order to improve the performance of BLE based proximity tracing using machine learning approaches. Most of the teams in this contest recognized that the underlying Bluetooth technology itself is not the best ground for accurate proximity estimation. Moreover, they noticed that the measured RSSI is highly dependent on the position or location of the phone itself, e.g., whether it is in a pocket or in the hands of a user. For the challenge NIST offered datasets collected by MITRE and the Lincoln Laboratory to the participating teams. The data included RSSI values that have been measured between various devices and other sensor data recorded by the phones, e.g. by magnetron or Inertial measurement unit (IMU) data. The evaluation of the models was done on unseen data, from which the normalized decision cost function (NDCF) was computed. The best results were achieved by the Contact-Tracing-Project-Team from Hong Kong University of Science and Technology [40].

A deep neural network was used to achieve the best results around a NDCF of 0.6 depending on the dataset and distance. As features for their deep neural network they used a histogram representation of the RSSI values collected, in order to reflect the multipath effects to the neural network. A second feature which was not used in raw format was the carriage state. This feature was extracted from the IMU data and the magnetron of the phone. Having the carriage state seem to make good improvements to the distance estimation, but the authors also claim that it is infeasible for a smartphone to compute these data with a deep learning model in time. Therefore, they presented a two-stage approach, where in stage one the IMU data is converted to fixed length vectors and stored to a database. The computational inefficient deep learning model on these IMU data vectors is then only called when the participant is at risk of exposure. Then the more accurate proximity will be computed [38].

Identifying advertisement channels at receiving time

The previously mentioned issue of not knowing which ADV_CH was used on the receiver side was investigated by Gentner, Günther, and Kindt. They developed a method based on timings to estimate the channel where a signal was received on. They successfully validated their approach on a range of android smartphones. On the devices where the approach is working, it has an accuracy about 100% in estimating the correct channel. Moreover, their approach can also be used on application level in android systems [34]. Having the precise channel information available, the exact frequency can be used to perform a more accurate path loss calculation.

2.11.3 Studies on proximity estimation using IEEE 802.11 signals

Except for the comparative studies, that compare BLE against IEEE 802.11 signals in terms of indoor positioning, almost no works were found on using IEEE 802.11 signals in terms of proximity estimation. The only one we found is the Comm2Sense paper by Carreras et al. [16]. Similar to other approaches, they presented a machine learning model to compute RSSI values, in order to estimate the distance. The values have been recorded between two devices, with having one device opening an IEEE 802.11 hotspot and having the other device connecting to this network. Then, messages are exchanged between these two devices and the RSSI is captured. They evaluated the approach in five scenarios with three different devices. The evaluation showed quite good results with a median error of $0.5m$ for the best devices.

In other cases, the error was smaller than $2m$ in 95% of the cases. Compared to the BLE results, these results were quite promising, but a counter argument against this approach named in several other studies was the power consumption by hosting an IEEE 802.11 network on a smartphone. Due to the fact, that this study was carried out in 2012 and some devices used do not support 802.11n via $5GHz$, it is very likely that in this study only the 2.4 GHz band was used to achieve these results.

2.11.4 Privacy issues of GEAN

Due to the fact, that adding IEEE 802.11 signals to the proximity estimation process, will make more meta-information of a user available, which can result in privacy implications, it will be good to observe the currently existing privacy issues of GAEN based application. Some studies that investigated on this issue will briefly be described in the following.

Formal definition and assessment of privacy goals in proximity tracing systems

Kuhn, Beck and Strufe defined several scenarios in a formal matter, that describe privacy issues, which can happen in a proximity tracing system. Based on this definition they evaluated several centralized and decentralized approaches. In their assessment, they separately evaluated server and client side. In their assessment for GAEN they stated, that a corrupted client can distinguish which people the corrupted client met are infected. This could happen when a client constantly logs the received RPIs with timestamps and the person met at that time. Then, it rebuilds the RPIs from the list of keys the backend published to determine which of the contacts was infectious. Additionally, by having a list of key of infected users, it can be determined how many users are or have been infected. The issue of determining the number of users that reported to be infected, holds for both server and client [43].

The power of the operating system vendors in GAEN

Jaap-Henk Hoepman from Radboud University argues that Google and Apple set-up a tool for mass surveillance with GAEN. By these two companies having full control on the proximity tracing, that is implemented in their smartphones, Apple and Google could abuse it, and they could be able to use this data for other purposes. This means that users and governments have to put massive trust in these two companies, to maintain and run the system as claimed.

Moreover, he criticizes that health authorities are not prevented by GAEN in setting up a fully centralized proximity tracing system, due to the fact, that the implemented applications by the health authorities keep track of what is being send to a central server [39]. Leith and Farrell have critiques that take a similar direction. They analysed several applications that are implemented on GAEN in terms of privacy. They state that the usage of *Google Play services* is the biggest issue in terms of privacy. Android users are enforced to activate these services. The play services regularly exchange information with Google servers. Even though, that Google states, that the users' identity is not shared with the company, users still have to trust in this statement [44].

Using the GAEN proximity tracing systems to derive location traces

Baumgärtner et al. placed BLE sniffers at some strategic places in a City. They impressively showed, that it is possible to create location traces, and even derive information for building an interaction graph, for infected users that are living and moving in the monitored area. When a user uploads his or her TEK to the backend, an attacker is able to derive the RPIs from that key, as explained before, and the attacker can then compare the RPIs of a user to the data collected, in order to build a location trace. When location traces of more than one user were identified, it is possible to compare these traces, in order to draw conclusions of social contacts and meetings that a person might had with people that were also declared as infected. Since the TEKs are only valid for one day, it gets harder to build such traces for longer time intervals, but they gave examples, which methods can be used to identify traces of the same person. In their paper, they also draw a scenario on how many BLE sniffing nodes would be needed to monitor most of the common used traffic ways for cars, commuter, and pedestrians in the city of Darmstadt in Germany. They argue that such a monitoring would be possible with less than 500 nodes. Additionally, they proposed and evaluated a relay-based wormhole attack in an experimental setup. With this setup they showed, that it is possible to sniff an RPI at location A and send it out again at location B, this allows sending out RPIs of infections people and different locations, for example to produce false positives. [10]

2.12 Summary

In this chapter, the background necessary for understanding the following parts of this thesis have been laid. Starting with information on contact tracing during a pandemic and defining the issues of manual contact tracing. Then, several digital contact tracing tools were introduced, with focus on proximity tracing solutions. The different methods of a GNSS based and a BLE based solutions have been compared and evaluated in terms of privacy. Moreover, the details of BLE and IEEE 802.11 broadcast messages were discussed, with a focus on signal propagation. Next, specific frameworks for a BLE based proximity tracing were presented and issues of such systems were discussed. In conclusion, it can be said, that the distance estimation using BLE signals with attenuation or RSSI thresholds is very challenging. This was made even more evident in the discussion of related work studies. Therefore, it became more clear, that an improvement of this existing approaches is needed. Such an approach for improvement, that utilizes IEEE 802.11 and BLE signals together, will be presented in the following parts of this thesis. Starting with the methodology and the explanation of an evaluation and measurement setup.

3 Methodology

In this section, the procedure and the approach are described in order to improve the proximity estimation and to answer the research questions. First, the approaches and ideas pursued to answer the research questions are described. Then, measurement goals will be derived from the approaches. Followed by the exact measurement setup and explanations of the scenarios where the measurements are carried out. Next, the procedure of ground truth data collection and the collected data itself are presented. Additionally, a procedure to calibrate the attenuation values received via BLE is discussed. Followed by an inspection of the measured data using 802.11 signals for bot bands, 2.4 GHz and 5 GHz. Ending a conclusion of this chapter.

3.1 Approach and idea

When looking at the research questions in the introduction chapter and looking into the related work, we see two approaches to improve proximity tracing systems. The first one is, finding a procedure of detecting the proximity in order to achieve better results than just using thresholds on a measured RSSI via BLE. This will mainly be the utilization of IEEE 802.11 probe requests together with the BLE advertisements. Once these probe requests have been sniffed, they need to be matched with the BLE signals. To use both signals for proximity calculations and this is the second part. Next, details on these two issues will be discussed.

3.1.1 Improving proximity estimation with multi-channel methods

As discussed in the related work section (2.11), the three BLE channels have different propagation characteristics. Some are more likely to get interfered by IEEE 802.11 signals [56]. Thus, having information on which channel an advertisement was received would probably help, to obtain more accurate proximity values. Our measurements with BLE signals showed significant differences in the measured RSSI values depending on the channel. These differences are visualized in figure 3.1.

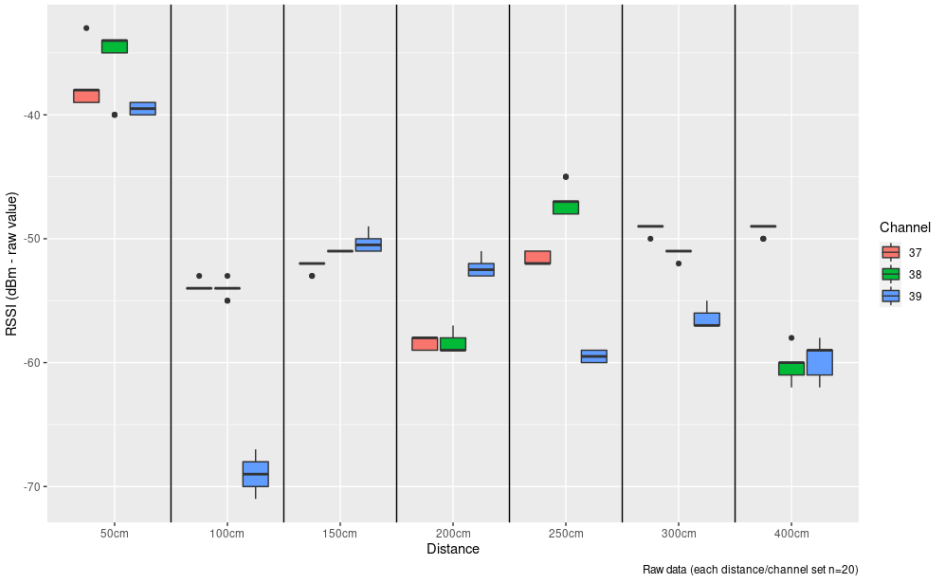


Figure 3.1: RSSI vs. distance, measured indoors in an office environment, grouped by the three BLE ADV_CHs

Details on the measurements will be given later in section 3.6, but for now, this result is just needed to clarify the issue of RSSI variance between the three channels, which are used. A big issue on differentiating between the channels is still, that the operating systems do not offer information on which channel the signal was received. Gentner et al. published a timing based approach in order to get this information [34]. Moreover, GAEN also faced this issue and then utilized the minimum attention for proximity estimation. Thus, the channel with the best signal strength is used and the variance by the other two channels does not influence the result, e.g., in an averaged attenuation value used for further calculation. Thus, it can be concluded that this issue was already addressed. Even though just taking the minimum attenuation may result in losing the information of the two other channels, when working with thresholds to determine the proximity, it is an approach that is good to use here. When using more complex models, e.g. propagation models that take the exact frequency into account, it may be sufficient to make use of the timing approach, mentioned before, to obtain the channel information. However, with using the minimum attenuation, the accuracy in proximity estimation got better, but it is still error-prone, as ground truth results will show later. Therefore, adding the channel of IEEE 802.11 signals into proximity estimation shall be evaluated as an approach.

With an IEEE 802.11 network interface put in monitor mode it is possible to capture probe requests across the various IEEE 802.11 channels on the 2.4 GHz and 5 GHz band. By capturing, both, the BLE and the IEEE 802.11 signals, more information, that was distributed across various frequencies and with different transmission power values, is available to describe or calculate the proximity. Moreover, for captured IEEE 802.11 packets, the information on which channel it was sent is always available. These additional packets and the information can then be used, to derive proximity thresholds, as it is already done for BLE in GAEN, or the information can be used to train a machine learning model. The development of machine learning models will not be in the scope of this thesis. However, some methods from the field of machine learning can be used, e.g., to extract the thresholds for classification. In first place, one might think that a typical regression on RSSI vs. distance could solve this problem. But as studies showed, especially in indoor scenarios the correlation between these features is too weak [45]. Moreover, the computational complexity of regression classifiers, like a support vector machine, is too high to effectively compute training data sets with multiple features, such as attenuation values for three signals and the channels that were used.

To achieve results that are comparable to existing related work and evaluation measurements, a classification approach based on thresholds will also be used here. Since not only RSSI or the resulting attenuation values are available, the channel information needs to be taken in account, too. Additionally, no thresholds are available for the IEEE 802.11 signals. To retrieve thresholds that result in accurate splits for classification, a decision tree will be used. The advantage is, that the splits are selected based on purity metrics, so the thresholds for certain features will result split a high number of samples split into the correct classes will be in the first layers of such a tree. With limiting the depth of such trees, it can be avoided, that the tree does over-fitting for certain environments. It is possible to train a tree on all features, so all signal types, or to train three different trees, one for each signal type, to get so-called specialized classifiers. The results of the specialized classifiers can then be combined for example in a weighted formula to obtain the overall result. Both of these methods will be evaluated. The specific parametrization, such as the depth of a tree, will be discussed in chapter 4 based on the results. Moreover, to benefit from the advantages of ensemble learners, that were described in the background chapter, the random forest algorithm will be evaluated as well, as a combination of various trees.

In order to obtain more accurate results, other sensors, such as the gyroscope or the magnetometer can be used as well. These can be used to detect if a person is moving to apply different models for various scenes. It is likely that using this information, combined with the new information from 802.11 will achieve even more accurate results. But due to time constraints, other channels and sensors will not be taken into account in this thesis.

3.1.2 Mapping IEEE 802.11 probe requests on BLE advertisements

In a laboratory setting it is quite simple to map recorded IEEE 802.11 probe requests on recorded BLE advertisements, to use both signals for proximity estimation. But assuming a real-life scenario and a possible extension of an existing proximity tracing system, like the German Corona Warn App, the complexity is rising. This is mostly because of the privacy preserving design of such an app, which makes it hard to determine which BLE advertisement comes from which device. Additionally, the MAC address randomization and the uncontentious sporadically sent probe requests make it even more challenging. Therefore, an approach is needed that makes the new information added by IEEE 802.11 signals usable, by matching it with the BLE signals. In general, to overcome the issue of MAC address randomization in IEEE 802.11 probe requests, approaches explained in section 2.7.3 can be used to re-identify a device over time. For example, the HT and VHT capabilities could be used for fingerprinting. Since only a limited number of people and no large crowds are near a probe sniffer, supported by the laws and rules that forbid such crowds during the pandemic, it is less likely that there are many devices of the exact same type in receiving range. Due to a limited number of devices and variety in the devices types, there is a reduced solution set. This makes it easier to overcome the MAC randomization issue, with simple techniques, such as the device capabilities fingerprinting. The limited number of clients around a receiver may also help to detect changes of MAC address and changes of the RPI for the BLE signals. But this is only needed for time intervals that are higher than the key change interval, of about 10 to 20 minutes. Specifics of the changes and the data sent, and an analysis of which behaviour and information may help to reduce the complexity of this mapping issue will be discussed later on the basis of the recorded ground truth data.

Conceivable approaches are now discussed and weighted in the following. Those that turn out to be promising will then be tested later in the evaluation with measured data.

1. Mapping on basis of RSSI values. This approach could be done in real-time, by just comparing the recorded RSSI values of the different channels received. Challenging in this approach is the differing transmission power and the device types. Especially the antennas, which result in varying signal propagation properties, can cause high deviations in the recorded RSSI values. This may be simple to implement for just one device, with having the transmission power information, but gets harder when more devices come in. Then, an accurate calibration is needed. Moreover, this approach is getting even harder by the various bands and frequencies used. A recorded RSSI value of a 5 GHz signal and a BLE signal sent via 2.4 GHz can hardly be compared. Depending on the environment, e.g., having a high reflective environment for radio waves, the variances get even worse. All in all, these facts make this approach less attractive, impractical, and prone to high error-rates.

2. Timing based approach, by taking the timestamps of both signal types into account. Such approaches have already been mentioned in related work ,e.g., [64], to overcome the MAC address randomization issue. But these studies utilized numerous IEEE 802.11 adapters, that enabled a sniffing on all frequencies, to get the timings inside a burst of probe requests across multiple frequencies. By having these timings, they were able to identify devices. But smartphones, which are used in the scenario of proximity tracing, do only have a single IEEE 802.11 network interface. Having only one adapter, it is only possible to sniff on one channel at a time. Moreover, since we want to have RSSI data from multiple frequencies, it is wanted that this single interface hops across the various frequencies in order to collect more data. This makes it even more difficult to take any timing into account, even if the timings between the bursts would be measured. Additionally, since the issue is not hard enough to solve, probe requests are not constantly sent like BLE advertisements. Depending on the smartphone they are mostly sent when the screen is turned on, to operating system specific intervals. Concluding this, a timing based approach in this case will probably be hard to take as well.

3. Generating device traces over time for both signal types, to match these traces, based on the time seen. As mentioned before, it is possible to identify a device across multiple random MAC addresses by fingerprinting certain fields of a probe request. By using this method, it is possible to follow a device over a certain time window. When this is also possible for the BLE probe requests, traces for a device can be generated and compared, by the time the devices are seen by the receiver, and additionally by the RSSI deltas to take the movements into account. For a time window smaller than the key rolling interval and the MAC address change interval, a single BLE MAC address can be used for this comparison. For longer inspection intervals it could be possible, taking the time a Bluetooth MAC address was seen first and last into account, to filter out all addresses that were seen in between. This could result in a very small solution set, depending on the number of devices in a certain area. Normally, the BLE advertisements are sent out in short time intervals of 5 to 10 seconds. Thus, it is likely that after one address appeared, and after all currently broadcasting addresses have been filtered out, because they have been seen before the change, that the address that is sending frequently after the change, is the correct MAC address. The IEEE 802.11 trace of MAC addresses and the trace of BLE MAC addresses may then be matched and used for proximity estimation. In terms of contact tracing, only contacts are needed that have been seen for at least 10 minutes or more. Thus, the approach of traces over time can be applied here, and no instant mapping is needed. When the time of devices seen is rising, more signal strength information and MAC address changes are available, to increase the probability of a successful mapping of two traces. The sporadically occurring probe requests on screen-on-time, do not challenge this approach, because the trace is build using fingerprints instead of the time of address change for probe requests.

Comparing the three approaches, approach three is the most promising in the use-case of proximity and contact tracing. This is because we have already tested methods available, that can identify MAC address traces for IEEE 802.11 probe requests. Furthermore, for contact detection in proximity tracing systems, time intervals of 10 minutes are used. With these 10 minutes, it is more likely to perform a successful matching of two MAC address traces. Thus, it will be implemented and evaluated on basis of actual measured data. Moreover, the first approach will be examined in presence of measured data as well. Additionally, when the first data is measured, the data will be analysed for other features that might help in reducing the number of possible solutions. Next, the goals of the measurements and from this following setup will be described.

3.2 Measurement goals

In order to evaluate the above approaches, data sets are needed on which the implementations can be tested. Some related work studies, e.g., the studies by Leith and Farrell [48] [47], uploaded their experimental data, but these do only contain BLE measurements. For this evaluation IEEE 802.11 data in 2.4 GHz and 5 GHz band is needed. This data should be sent with the same device in the same environment, to give statements on how the approach will perform in a real-life environment. Therefore, a measurement campaign that collects BLE and the extra IEEE 802.11 data is carried out. This measurement campaign will additionally fulfil the following goals:

- Ground truth data captured in a LOS setting, for each signal type with increasing distances. The distances should address the three distance classes defined by the national contact tracing applications.
- Each measurement iteration should last long enough, to avoid any short time effects, such as higher noise level.
- Different scenarios and environments need to be measured. Otherwise, it might be possible that the thresholds obtained from the data, work perfectly in the measured environment, but perform badly in other environments. Having measurements taken in multiple environments will avoid over-fitting, too.
- The collected ground truth data and the scenarios should be comparable to the measurements from related work. This can then be used for evaluation purposes.
- Various device types should be used as a sender, to evaluate whether the performance and accuracy are device specific. Especially the antenna design, and differing transmission power rates are likely to cause better or worse signal strength values.
- A monitoring of noise on the observed frequencies should be used, in order to detect disturbing influences.

Fulfilling these points, the measurements should give a good base, in order to evaluate the implementation of the proposed approaches. A first evaluation of the measurement setting itself can already be done after collection of the first ground truth data in BLE, by comparing it to related work studies. The accuracy values of distance estimation can be calculated using the BLE RSSI values, and then be compared. When these results are similar, further measurements in additional environments and scenarios can be carried out and evaluated. In the next section, a measurement setup will be presented that takes the previously mentioned points into account.

3.3 Measurement setup

For the measurements, two different setups are used. One for measuring the ground truth data, and a second one to measure real-life scenarios. The main difference is, that in the ground truth setup, only one device is sending out signals, with a constant device orientation over all iterations. In the real-life scenarios multiple devices will be used for sending, the orientation and position of a device may vary, and there may be no LOS in some settings.

In all measurements, a *Raspberry Pi 4b* will be the receiver. This decision was taken, because it is easy to develop and deploy sniffing scripts on the Linux system running on it. Moreover, the Raspberry has a *Broadcom bcm43455c0* wireless chip. This chip supports the monitor mode kernel patch[1] so that IEEE 802.11 probe requests can be captured.

The BLE sniffer was written in Rust, due to its' high performance, capabilities to access hardware interfaces and its' built-in memory safety. The *crate bluez*[2] (library) was used to implement the sniffer. Using this crate, it was easy to implement a sniffer. In proof-of-concept work, other libraries like *blurz*[3] have been tested, but not all packets were sniffed properly. Using *bluez*, the implementation and the communication with the Bluetooth interface was more easy and more stable. At least the post-processing, like extracting flags, addresses and the payload was also easier using *bluez*. The sniffed packages are then written into a *SQLite* database, with a timestamp, RSSI, MAC address, length and the payload for each package. The *SQLite* allows during the measurements to already analyse the data, filter and apply functions such as average. To sniff the IEEE 802.11 probe requests, *tshark*[4] is used, a program to dump and filter traffic in *pcap* format. These files can then be analysed in tools like *wireshark* or easily computed using the Python *scapy* library. Moreover, a tool to trigger channel hopping for the IEEE 802.11 interface is needed. Otherwise, only a single channel is monitored. Therefore, *airodump-ng*[5] from the *aircrack-ng suite* is used. This tool can also dump packets as a pcap file by itself, but it does not offer a variety of filters like *tshark* is doing that. An essential filter that will make the analysis easier, for example, is a frame type filter, so that only management frames from the type probe request are captured. This reduces the amount of data and enables faster computing of the pcap files later.

[1] `https://github.com/seemoo-lab/nexmon`
[2] `https://github.com/laptou/bluez-rs`
[3] `https://github.com/szeged/blurz`
[4] `https://www.wireshark.org/docs/man-pages/tshark.html`
[5] `https://www.aircrack-ng.org/doku.php?id=airodump-ng`

Next is the transmitter side. In order to fulfil the goal of device diversity, three devices are used for transmission:

- OnePlus Nord N10 5G
- Apple iPhone 6S
- Raspberry Pi 4b

The measurements will be carried out with each of this three devices. The selection ensured that, an Android device and a iOS device are used. Both smartphone models support GAEN and the national proximity tracing app deployed in Germany. For constantly sending out probe requests the automatic screen turn-off is disabled on both smartphones, and the Wi-Fi menu needs to be constantly opened. For sending out BLE advertisements the App *nRF Connect*[6] is used in the ground truth scenarios, and the *Corona Warn App* in all other scenarios. On the sending raspberry the *bluez sample advertising script*[7] is used for sending out BLE advertisements. To send out probe requests the CLI utility *iw*[8] is used to trigger IEEE 802.11 scans, for constantly doing so a simple while loop in a shell script with short pauses is used.

To observe any interference or noise the spectrum of 2.4 GHz and 5 GHz is sporadically observed with a spectrum analyser. Therefore, a *Metageek Wi-Spy DBx* is used in combination with *spectools* for visualization purposes[9]. Next the specifics of the ground truth measurements will be explained.

[6] https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Connect-for-mobile
[7] https://github.com/bluez/bluez/blob/master/test/example-advertisement
[8] https://wireless.wiki.kernel.org/en/users/documentation/iw
[9] https://github.com/caljorden/spectools

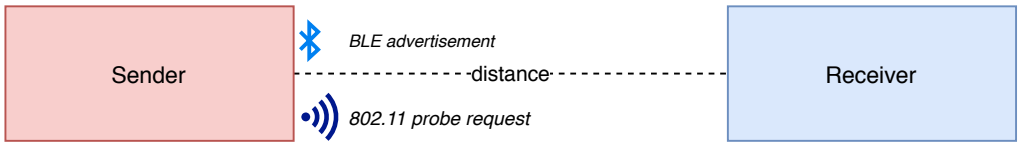


Figure 3.2: Schematic measurement setup to collect ground truth data

3.3.1 Ground truth setup

For the ground truth measurements a minimalistic setup will be used, with having a receiver and just one sender active in the direct environment. A schematic visualization can be seen in figure 3.2. A sender, is sending out BLE advertisements and IEEE 802.11 probe requests. To lower the side effects of multi-usage of the interface and queuing, only one of these signal types should be measured at a time, if possible. Sender and receiver are placed facing each other, with a distance d in-between them. For every device and environment d is measured ranging from at least $50cm$ to $400cm$, increasing d by $50cm$ for each iteration. This ensures having *very close* samples for the distances $50cm$ and $100cm$, *close* samples for $150cm$ to $300cm$, and *safe* samples for $350cm$ and $400cm$. If it is possible in the measurement room or environment higher d than $400cm$ will be captured as well. An exemplary measurement section in an indoor office environment, having the $50cm$ steps marked with tape, is visualized in figure 3.3. Sender and receiver are placed at a height of about $50cm$ to have a comparable setting for a scenario like sitting in a tram or bus, as it was used in [46]. The devices are put in front of a chair to have a low reflection by the chair itself. Moreover, a little height as the $50cm$ over the ground level, will have a later interfering ground reflection, than directly placing the device on the ground. For each distance measured, BLE data and IEEE 802.11 data will be captured. Initially, measurements that run for several hours are needed, to examine whether there are any peaks or effects over time. These effects can for example, be caused by business hours, so when more people having devices communicating on the measured bands are in the building. If this examination does not show significant changes over time the time periods, the time a measurement is running can be reduced, in order to carry out more measurements in a shorter time. These details will be discussed in section 3.6, including the presentation of the measured ground truth data. When the ground truth data for an environment is collected, certain scenarios can be measured, the setup used for these scenarios will be described now.

3.3.2 Scenario setup

In real-life scenarios, the same measurement infrastructure, as used in the ground truth setup, is used. The main difference is, that more devices are sending out probe requests and BLE signals at the same time. Moreover, in this setup the BLE signals from the phones are sent out via the GAEN based app, in order to generate data, to examine the approaches for mapping probe requests to BLE advertisements. The distance for each of the three senders to the receiver can vary and obstacles, like a human body, can be in the LOS in these measurements. The schema of the setup is visualized in figure 3.4.

Figure 3.3: Exemplary measurement section in an indoor office environment

Additionally, the two signal types are sent at the same time, to come as *close* as possible to everyday conditions and use. Depending on the environment other signals of random unknown devices can be captured. This data will make the mapping evaluation more challenging. As described before, no personal data and only pseudonymous data by the random MAC address and RPI is logged for both signal types. The IEEE 802.11 sniffer is configured to only capture the probe requests and no other data traffic, that might be unencrypted and will enable de-anonymization. The same holds for the BLE data captured, other advertisements not having the service UUID of GAEN will be deleted. In the following, the scenarios and environments where the measurements have been carried out will be presented.

3.4 Measurement environments and scenarios

Several scenarios needed to be measured in order to have a solid base of data to evaluate the approaches. In total four scenarios have been measured. Including difficult environments such as a bus and a train, with a cabin completely built out of iron and high reflections for radio signals. The train and *bus* environment have been especially chosen, to create comparability to the studies of Leith and Farrell [46] [45] [48]. In initial plans, only three environments where on the list for measurements, but due to some issues in the office set, a second set, the meeting set, was added in replacement. Both were selected to have a typical office indoor environment. The details and an analysis on the issues with the office set are explained in Appendix 6.4. In the following, each environment and the measurement scenarios carried out there will be described.

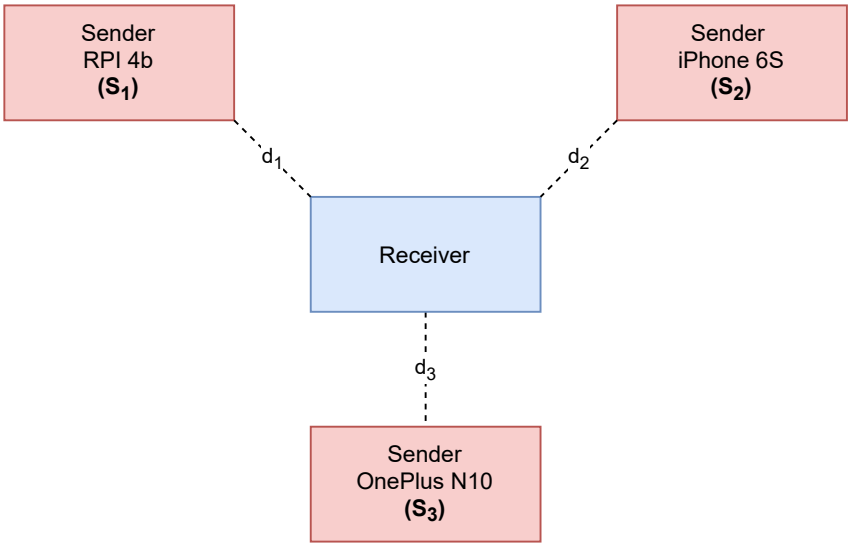


Figure 3.4: Schematic measurement setup to collect scenario data

3.4.1 Meeting room environment

This room is in the same building as described for the office environment set, but it is on the other side of the building. A monitoring carried out over several days, showed that the noise peaks do not exist in this room. Therefore, the indoor measurements have been carried out in this room again. This time with consistent devices, as described in the setup subsection before. Due to the fact, that this room is a bit smaller than the previously mentioned one, the distances for the measurements only ranged from $50cm$ to $400cm$. The measurement lane used, consisted of two chairs with increasing distance, such as in the office scenario. Additionally, some meeting settings have been measured in this room, with having the 4 clients distributed around a table. In this setup, the devices have been placed at the edge of the table. The setting is visualized in figure 3.5. The sending devices were switching the positions after one hour of measurement, the receiver was always at the same place. To test against over-fitting, similar scenarios have been carried out in another meeting room in the building, with a higher distance up to $400cm$ between a certain sender position and the receiver, the setup was almost the same, with devices on tables as in figure 3.5. Since these are *out-of-plan-measurements*, in the same building, at the end of the campaign, this data was added to the scenarios of the meeting room set.

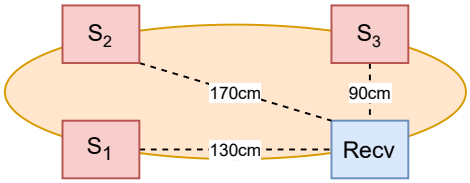


Figure 3.5: Scenarios measured in the meeting room environment

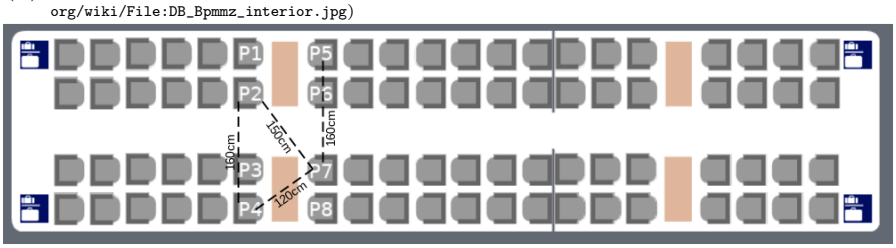
3.4.2 Train environment

The measurement data captured in the train on a journey from Osnabrück to Berlin and back. It was an Intercity (IC) train operated by Deutsche Bahn (DB). The wagon of the train was the overhauled and modernized version of the Intercity wagons, a photo of the wagon type is presented in figure 3.6a. For each way, two measurement scenarios of about one hour have been recorded. For this, two seat groups of four have been used to place the devices. A seating plan with distances is visualized in figure 3.6b. In the first two scenarios, the receiver was placed on $P2$, and persons were sitting on $(P1, P6, P3, P8)$. The other seats have been used for the sending devices. On the way back, for scenarios three and four, the persons were sitting on the same seats, but in these scenarios, the receiver was placed on $P4$. The wagon was in both scenarios not very crowded, about 10 other persons have been around. Some other travellers also had an activated GAEN based proximity tracing application, the data was also collected to test the mapping approach. Due to the time limitation of the journey, only these scenarios have been measured in the train, without measurements on a measurement section. The highest distance measured was from $P4$ to $P5$ with $250cm$. The scenarios measured on the way back had an issue for the IEEE 802.11 channel hopping, only channel 36 (5180 MHz), was captured in these scenarios. Because of the pandemic situation no other measurements in public trains have been carried out until now.

3.4.3 Bus environment

The data set collected in a bus was the last one of the measurement campaign. An empty bus was provided by Osnabrück's local bus provider *Stadtwerke Osnabrück*. They provided an empty bus on their premises. The bus used was a modern *VDL Citea SLFA-181* electric bus. Two days were allocated for measurements. During these two days, all devices have been measured on a measured section inside the bus. Distances from $50cm$ to $400cm$ were measured. Additionally, the distances $500cm$ and $600cm$ were measured in this environment. The measurement section is presented in figure 3.7b. Except for the ground truth data from the measurement section, three scenarios were measured in the bus. The seat plan of the bus is visualized in figure 3.7c.

(a) Wagon interior overview (Cherry25565, CC BY-SA 4.0 via https://commons.wikimedia.org/wiki/File:DB_Bpmmz_interior.jpg)



(b) Plan of seats and measured distances (illustration by https://db.de)

Figure 3.6: Train cabin used for measurements

The first scenario was measured in a seat-group of four at *very close* distances. With the receiver placed on $P2$ and the senders on $P1, P4, P5$. For the second scenario, longer distances have been used. For this, the sender was placed on $P1$, the OnePlus on $P10$ with a distance of $420cm$, the iPhone on $P11$ at $520cm$ distance, and the Raspberry Pi on $P7$ at $230cm$, with a person sitting on $P6$. In the third scenario, *close* distances were measured. The receiver stayed at $P1$, the OnePlus was placed on $P3$ at $195cm$, the iPhone on $P9$ with a distance of $260cm$ and the Raspberry stayed on $P7$ at $230cm$. Each scenario was measured for at least half an hour in the *bus* environment.

3.5 Device specific calibration

For the GAEN implementation, Google is having a list of calibration values for transmission power and calibration values for RSSI, because these can heavily vary from device to device [7]. To the time of writing this thesis, no calibration values were available, for the devices used in the measurements. And the proposed procedure for calibration by Google, could not be used, because the newest *Google Pixel* smartphone was needed to do the calibration. But to use the attenuation thresholds of $55dB$ and $63dB$, which are also used in the German proximity tracing application, a calibration is needed. Otherwise, values will always be too far from

(a) Outside (b) Measurement section (c) Seat plan

Figure 3.7: Bus used for measurements

the defined thresholds. Another way would be the definition of own thresholds, but to have comparable results, this approach is not considered, except for an evaluation step using one classifier for all three signal types.

To obtain calibration values, constrained programming is used. Using a constrained programming model, equations on calculating the attenuation can be formulated, including a variable for the calibration factor. Then, a constraint can be set, that for a certain RSSI raw value, the attenuation result should be less than the thresholds. The RSSI values used in the constrained program, are represented by the average over the one-minute maximum RSSI. This value is used because the minimum attenuation values in GAEN are also used for the risk assessment. More details on why we use a maximum RSSI will be given in the next section by looking at the measured data. In addition, it then becomes clear why some constraints had to be relaxed in order for the programme to be solvable at all. For this relaxation a decision was needed, whether to take a trade-off in more false-positives or false-negatives into account. We decided for having more false-positives, because in a pandemic situation it is more severe having a false-negative, so a person that is infectious but not getting tested because no alarm was given, instead of having a false-positive, so a person that is not infectious, that is alarmed. These false-positive can then easily be resolved by testing the person. Of course, this trade-off can only be used, when there are enough capacities for testing available. To model and solve the constrained program, the python package *ortools*[10] is used.

[10] `https://developers.google.com/optimization`

This is an open-source programming library, developed by Google, for modelling and solving optimization problems. The model used for obtaining the calibration value for the OnePlus is presented in the appendix in listing 6.1. Three variables are used, c_2 is the wanted calibration factor, sep_2 is a variable for the optimization function. sep_2 is used to find the smallest RSSI value that can separate the *very close* set from the *close* set. For the same purpose, sym_2 is used, the difference is that this value is used for separating *close* from *safe* distances. The equation used to model the constraints is the same equation that GAEN is using for calculating the attenuation, the -11 in the beginning is the transmission power value of the OnePlus. Since this is an integer optimization program, the input RSSI average values, are also needed as integers. Therefore, the numbers have been cut after the comma, instead of rounding. So the numbers are smaller than in reality. To take this into account, the cut $1.5m$ and $3m$ values are modelled in constrains smaller than the threshold, instead of greater equal. We chose this way, to push the trade-off in the direction of producing more false-positives in the classes for *very close* and close. Moreover, to achieve this, the end for *safe* classes is open, and the next possible RSSI value, that does not conflict with the other constraints is approximated with thy sym_2 variable. The maximum RSSI values used in this model are obtained from the meeting data set, since this is the only data set in a relatively *normal indoor environment* and this was one of the data sets that was measured first. In order to continue with other packages of work, which were dependent of a calibration, only these values from the office set have been used. However, we assume that using a single data set for obtaining device specific calibration values, is the best way, because we want to describe the device specific differences and not scenario specific differences with such calibration values. Using means across data samples measured in various environments, also environmental influences would be modelled with this calibration.

The calibration values that resulted from using this procedure are presented in table 3.1. As mentioned, it is intended, that these calibration values may result in more false-positives in the closer proximity classes. The quality of this calibration values will briefly be discussed in the following section on ground truth data.

Device	**Calibration value**
OnePlus Nord N10 5G	2
Raspberry Pi 4b	14
Apple iPhone 6S	15

Table 3.1: Attenuation calibration values obtained by the constrained programming solver

3.6 Ground truth data

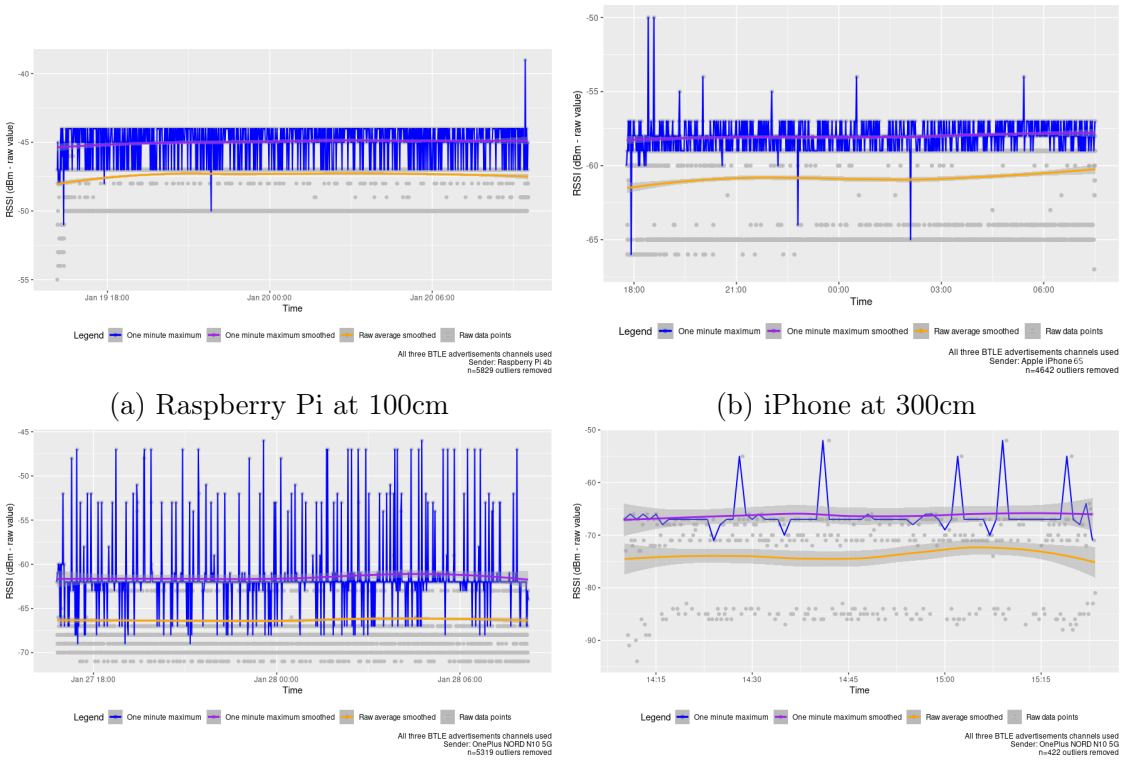
In this section, the ground truth data that was collected during the measurements will be presented. The data itself will be presented. Starting with the BLE data, including a comparison to related work studies. Followed by the IEEE 802.11 data, separated by frequency bands, 2.4 GHz and 5 GHz. Moreover, the ground truth data was analysed for certain characteristics, that helps for later implementations, will be described in the subsections of the regarding signal type.

3.6.1 BLE data

The BLE ground-truth data measured will be presented in the following. At first, we will present BLE RSSI data for all three advertisement channels measured over time, to discuss if there are any time based effects, in order to assess which time is needed for a measurement iteration. Then, the issues of using three different channels are discussed, based on the measurements. Moreover, the approach of using just the minimum attenuation measured in a scan window, as GAEN is providing it, will be evaluated. Next, the differences of the attenuation values measured in the three different scenarios will be presented. In the end, performance metrics for a classification only done on BLE attenuation values will be discussed for the three devices used.

Measurements over time in detail

In figure 3.8, four plots are presented. They visualize the measured data points as grey dots. Figure 3.8a contains the data samples measured with the Raspberry Pi 4b as a sender on $100cm$, the measurement was running for more than 16 hours. The plot shows a clear distribution of the data points into three groups, the one with the highest RSSI value is a bit hidden behind the one-minute maximum line in blue. These three groups at RSSI values of $-45dBm$, $-47dBm$, and $-50dBm$, are that clearly divided, because these are the three BLE advertisement channels, that were used for sending. Except for a little noise at the beginning of the measurement, probably because of the person in the room that started the measurement, the RSSI for each channel remained constant. The constant raw average line in yellow and the purple smoothed one minute maximum proof this statement. The variance around the maximum is usually not higher than $2dBm$ in this plot. Time measurements for each device 3.8. Next in figure 3.8b, a similar observation can be stated. This plot shows the data collected for the Apple iPhone 6S at a distance of $300cm$. Again a separation of samples into three groups, having constant average lines over time.



(a) Raspberry Pi at 100cm

(b) iPhone at 300cm

(c) OnePlus at 300cm

(d) OnePlus at 200cm

Figure 3.8: Measured RSSI data for serveral devices over longer time windows

The variance around the one-minute maximum does also take about $2dBm$ into account. The difference between the channel with the best RSSI value and the one with the worst value is in this set with $8dBm$ a bit higher than in the set from the Raspberry Pi. Next, in figure 3.8c the data measured from the OnePlus Nord N10 at a distance of $300cm$ is presented. By just looking at the blue one-minute maximum, we can see that the variance around the maximum is higher than in the other sets, seen before. In some peaks, the signal is by $12dBm$ better. Moreover, two channels seem to have *close* measured RSSI values between $-67dBm$ and $-70dBm$. We assume that there are some distances for each device where a higher noise, probably caused by the ground path, comes into the game. This can result in such a variance around the maximum. However, the pattern stays the same over the 14 hours of the measurement. That this high variance around the maximum is caused by the distance and not by the device type itself, can be described with a measurement for that device at $200cm$, as presented in figure 3.8d. In this plot, the blue maximum line is almost constantly *close* to the purple average line at $-78dBm$, with a constant pattern over the hour of measurement.

The observation that certain distances in the interval measured here may cause higher noise by the ground path, can be substantiated by computing a Two-Ray ground-reflection model, using the parameters from the measurement. We computed such a model implemented in python [66], to further investigate this issue, and it supports our observation. The plot of the model is visualized in figure 3.9. The blue line shows a minimum for a distance of $3m$ to $4m$, where the normalized path loss is very high. Of course, this model is just a simplified description of the issue, and the limitation is, that specifics of the antenna, the transmission power, and details of the environment itself are not taken into account in this model. But this simplified model helps to understand the issues of ground path interference in indoor environments better. Going back to the plots in figure 3.8, three of these plots are based on the longest measurements in the meeting data set. As we have presented, the pattern of the three channels and the RSSI values are constant over time. Similar observations were made on other distances measured in that environment. Additionally, in the not used office data set, measurements over several days have been carried out, except for the peaks caused by the interference that was described earlier, the same observations were made. It turned out that averages calculated on the RSSI values from a complete data set, and just on the one-minute maximum RSSI, do not vary significantly after 10 minutes of measurement. Thus, the measurements do not necessarily have to be carried out over several hours, which was helpful for the measurement of the *bus* environment, where the vehicle itself was only available for a limited time.

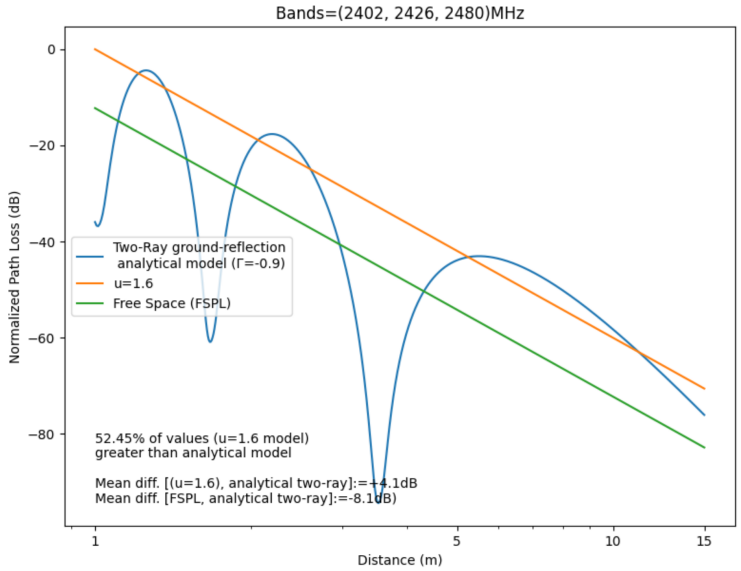


Figure 3.9: Analytical Two-Ray ground-reflection model for BLE advertisement channels for a height of 47cm (Calculated with model [66])

.

Variance caused by using three advertisement channels

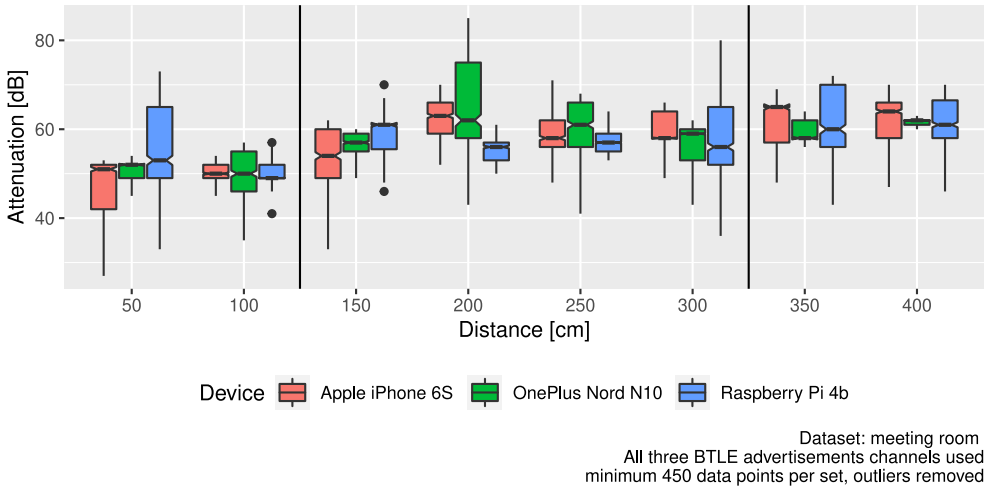
To briefly assess the change made in GAEN v1.5, where the possibility was implemented, to use the minimum attenuation of a time window instead of an averaged value over all captured signals, we will show two graphs to visualize the impact of this change. In figure 3.10a a box plot for all data samples collected at a certain distance for a specific device is presented. Except for the observation that there is no steady increase of the attenuation in dependence of the distance, we can see very long boxes in the distances of $50cm$ or $200cm$, with a 1st and 3rd quantile that range wider than $15dB$. This indicates a high standard deviation and variance on the measured data. Such large variations in attenuation make it almost impossible to accurately classify distance using thresholds. For example, the plot for the Raspberry Pi at $50cm$ is ranging across all thresholds, below $55dB$ for very close, below $63dB$ for *close* and also above $63dB$ indicating a *safe* distance. The mean is at about $57dB$, which will cause that this would be classified as a *close* distance, instead of the true class, the *very close* distance. Similar issues can be observed for the other distances and devices as well. When we compare this to figure 3.10b, here the one-minute minimum attenuation for this data set is visualized. The boxes in this blot are much smaller than in the plot for all samples before. The variance in some cases is so small, that only the average line is plotted. By applying a minimum-filter, it is very likely that only data of a single channel, the

channel with the best propagation characteristics in the current situations, will stay in the set. To verify this observation, we configured the Raspberry Pi 4b to only use a single one of the advertisement channels and measured the data for all three channels each for the same distances. The measured data showed, that the variance of the data samples is also very small, comparable to the plot in figure 3.10b. The results of the single channel measurements were already presented before in figure 3.1. By having these observations, we can state, that changing the measurements to using the minimum attenuation instead of the average, resulted in fewer variant data, that always represents the best channel. This will then result in a more accurate distance estimation. Based on the measured attenuation value, this is closer to the optimal attenuation for a given distance.

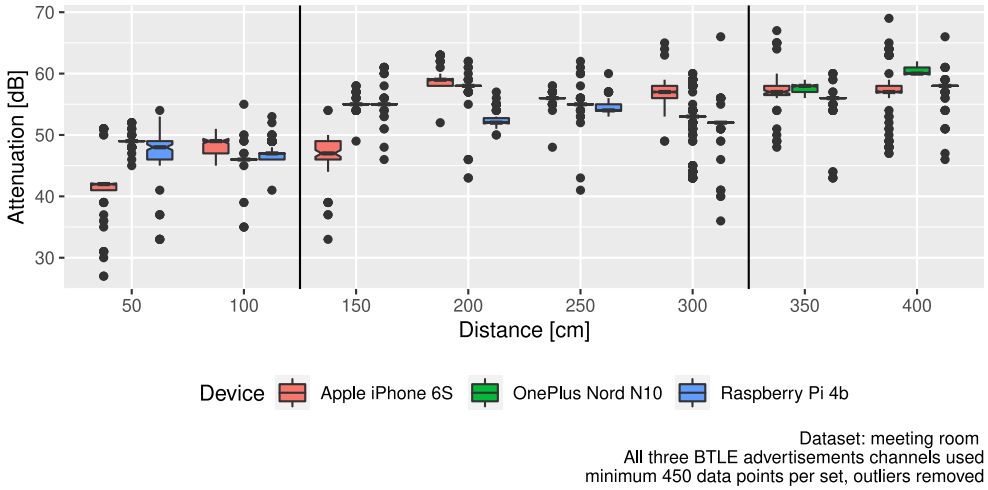
Another observation can be made while looking at figure 3.10a, for all three devices. The measured averages are *close* together and the distance between the averages of the three devices is never higher than $5dB$. It behaves quite similarly in figure 3.10b for the one-minute minimum attenuations. This indicates, that our calibration approach, presented in section 3.5, is capable to establish comparable attenuation values for the three devices used.

Comparison of the three measured environments

Next, we will compare the measured ground-truth data for the different environments. For this purpose, we created one plot for each device. The plots are visualized in figures 3.11a, 3.11b, and 3.11c. In these plots, the average minimum attenuation against distance, for each measurement environment is visualized. It needs to be mentioned, that the interpolation between the data points will not necessarily reflect the values that will probably be measured in between two data points. We just use this connection of the actual data points, to obtain a better overview. We will start with Figure 3.11a. It shows the plot for the OnePlus. The environment with the lowest attenuation values is clearly the meeting room. Problematic for classification in this environment is the local minimum at 300cm. This local minimum is probably a result of ground-path interference for distances around $300cm$, as described earlier. Next, is the *bus* environment. Compared to the meeting room, the *bus* environment seems to have a higher base attenuation. The same applies to the *train* environment. But, as we will see later, this observation can only be made for the OnePlus with its' device and antenna characteristics. When looking at the *train* environment for the OnePlus, the measured values for $160cm$ and $250cm$ are above the $63dB$ threshold (dotted yellow line), which means that these will also be classified wrong into the *safe* class, instead of the correct *close* class.



(a) Box plots for all three BLE channels



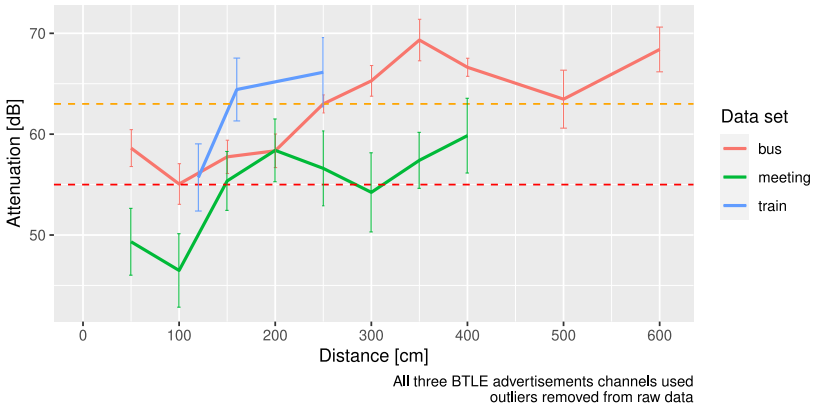
(b) Box plots for minimum attenuation

Figure 3.10: Comparison of complete BLE data samples measured against the one-minute minimum attenuation

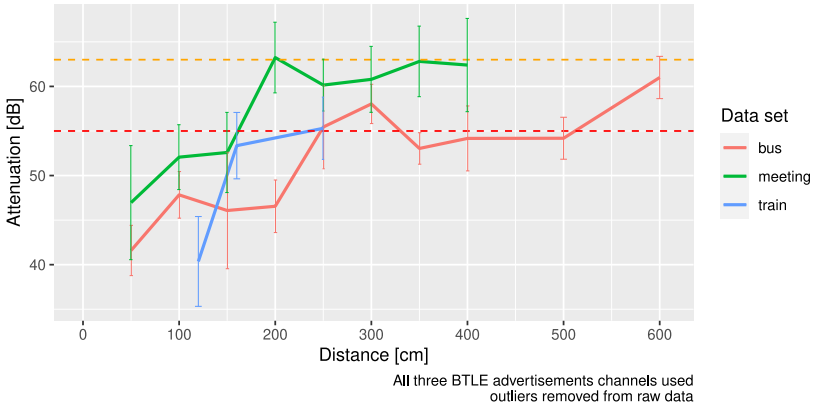
The plot for the iPhone 6S is in figure 3.11b. We can observe that the data for the *train* environment is almost completely below the threshold, which means that it would be classified as very close, which would be wrong for $160cm$ and $200cm$. It behaves similarly for the *bus* environment. Another point to mention is, that the minimum attenuation for the iPhone is, except for some points in the *meeting* environment, completely below the $63dB$ line. This will result in very few classifications for the *safe* class. For the measurement in this environment, we can also conclude, that there are problems in separating *very close* and *close* distances for the bus and *train* environment. This is different for the *meeting* environment. Here the separation between these two classes seems to be better.

Ending with the last plot of this type, for the Raspberry Pi 4b in figure 3.11c. Compared to the other plots, the attenuation values measured here are shifted down. None of the lines is even touching the yellow $63dB$ threshold line. The antenna design for the Raspberry Pi sender is most different. With facing directly to the receiver, it is very likely that this is the reason for the better attenuation values. Looking at the line for the *bus* environment, this effect may also have a strong influence, even for distances of $600cm$ the attenuation is not increasing, and it is lower than at smaller distances. In the line for the meeting scenarios, we can observe the issue of calibrating at $150cm$ threshold, which is slightly above the red threshold line for $55dB$, but then gets below this line at the next higher distances from $200cm$ to $300cm$. Since the calibration that is used here is affecting all sets measured by adding or subtracting a calibration factor, a higher factor would not necessarily help to solve the issue, a huge part of the *safe* distances in the *bus* scenario would still stay in the wrong class. The same holds true for the data points from the *train* environment, where the attenuation at $150cm$ is lower than the one at $50cm$ in the *meeting* environment.

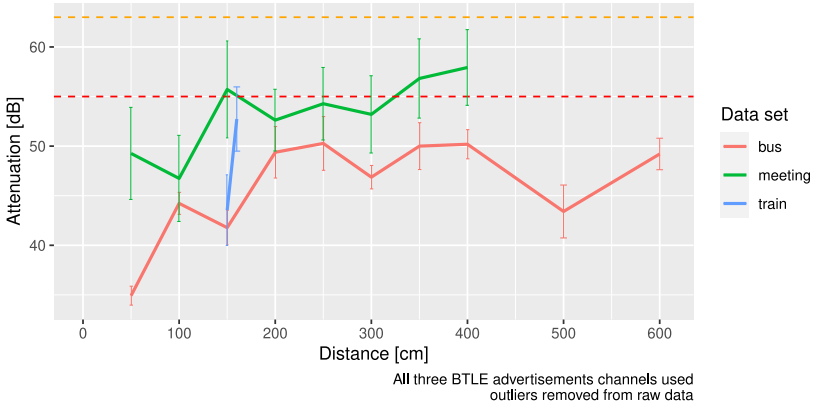
All in all, the three environments and three devices show significant varying curves. It is difficult to detect any propagation pattern. Even for a single device, the measurements for the three environments show that the propagation characteristics for each environment can heavily vary and make it very challenging to estimate the distance from the attenuation level. Comparing these curves with the single channel measurements in indoor scenarios by Nikoukar et al. [56], it can be stated that these curves are comparable, and they had similar observations. Especially the peaks in-between where the attenuation gets suddenly lower at a certain distance and then worse again. Additionally, the standard deviations, presented in the error bars are also comparable to each other in indoor environments.



(a) OnePlus Nord N10



(b) iPhone 6s



(c) Raspbperry Pi 4b

Figure 3.11: Comparison of the mean minimum attenuation measured per scan window in the various environments per device

Nikoukar et al. describe the issues observed in the indoor measurements with shadowing effects and high numbers reflected signals that overlap at the receiver and cause higher attenuation. Moreover, they state that any small variation in the position of the sender or the receiver may result in a significant change of the wireless channel. Finally, interference by IEEE 802.11 signals around is another important issue that can strongly increase the attenuation of a BLE signal.

Classification performance on the BLE ground truth data

In the following part, we will present accuracy results, that were calculated on the BLE ground-truth data. The minimum attenuation per scan-window was used, as it was presented in the visualization and analysis before. For each sample in the set, the accuracy is compared to the threshold and a class is assigned based on this comparison. Used thresholds are less than $55dB$ for the class very close, less than $63dB$ for *close* distance, and all above is classified as *safe* distance. The data is grouped per environment. Starting with the meeting room evaluation in table 3.2. In this table, a sub-table represents the performance metrics for each device. The commonly used metrics from the statistics and machine learning sector are used. On top of the sub-tables the confusion matrix with the manually classified, so true samples per line and the by attenuation classified samples per row. The diagonal line in the table marked in green here represents the true positive. These are the samples that have been assigned to the correct class. The cells above and below a green cell, are false-positives, samples that have been assigned to the class of the current row, but belong to a different class. Left and right from a green cell are the false-negatives. These are the samples that have been falsely assigned to another class, but would belong to the class of the row of the current viewed green cell. Below the confusion matrix accuracy and the macro F_1-score for the complete set are printed. Below the global quality metrics, the precision, recall, and F_1-score for assessing the quality of a certain class are listed. For this set the classes are unbalanced, this means that per class the number of samples may vary. For example, this is already the issue in table 3.2a. In this set, there are about 300 samples each for the *very close* and *safe* class included and 2202 for the *close* class. This can bias the accuracy value when the class with a big amount of samples is better or worse than the others. So the accuracy may not be a good metric in these tables. Because all of these sets are unbalanced, this results from assessing the performance directly on the not sampled, measured raw data. The metric that will give a better statement on the classification quality for the complete set is the macro-F_1-score. It is the sum of f1-scores per class divided by the number of classes. Using this, the quality of each class is going to some parts into the global value.

The OnePlus results, listed in table 3.2a, show that many samples have been falsely assigned into the *very close* class. This results in a very bad precision value, with 0.18. The precision indicates the ratio of the correct assigned samples of the total amount of samples predicted for a certain class. The recall is at 0.99 because only four samples out of the real *very close* samples have been assigned to another class. The F_1-score with 0.31 indicates the poor performance for this *very close* classification. Next, the *close* class has a better precision with 0.75, but the recall is quite bad, because of the many samples that were assigned into very close, which should actually belong to the class close. With f1 at 0.54 classification performance in this class is poor, too. Even worse it is for the *safe* class, with only 13 samples predicted for this class and 5 of them correctly. Concluding this over all classes, the F_1-score is at 0.29, a classification with a very poor performance. The accuracy, even when it is biased, only 44% of all samples have the correct class label. For the Raspberry Pi 4b the situation is different, the performance for this device is better than for the OnePlus, but still with much room for improvement. In this set, presented in table 3.2a, the *very close* class was well predicted, with a precision of 0.81, a recall of 1.0, resulting in an F_1-score of 0.9, which is an astonishing result for such a classification. The other classes had similarly poor results to the OnePlus. In this set, the biased accuracy can be observed well. Because of the imbalance of the size for the different classes, the accuracy is pushed to 0.77. This results from an over representation of the *very close* class. Next, the Apple iPhone 6S in table 3.2c. The *very close* prediction with an F_1-score of 0.7 is quite good, too. The best predicted class in this set, is the *close* class, with a precision of 0.83 and a recall of 0.78, resulting in an F_1-score of 0.81. The *safe* class is again problematic, but slightly better than for the other two devices. Overall, the set has a macro F_1-score of 0.63, which is the best performance among the three devices in this environment.

In the performance results, the prediction quality of the *safe* class was the worst. This may be a result of the calibration, which was done using this set. As explained in the calibration section, the constraints for the calibration in the *safe* class needed to be relaxed, in order to obtain a solvable system of equations. Therefore, and in order to have a trend towards false-positives rather than false-negatives, this trade-off of having a worse *safe* distance classification was accepted.

Further on, the quality of the results from the bus, in table 3.2, will now be discussed. Again all three devices have been measured in this environment. The times for each measurement iteration were a bit shorter, which explains the smaller sample amounts. The set presented here is again $\frac{1}{3}$ of the total number of measured samples, because it is the minimum attenuation of a scan window, so the minimum of three values.

OnePlus Nord N10 5G				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	288	4	0	292
close	1259	935	8	2202
safe	21	301	5	327
***n*-pred**	1568	1240	13	2821
				accuracy: 0.44
				macro F_1-score: 0.29
precision	0.18	0.75	0.38	
recall	0.99	0.42	0.02	
F_1-score	0.31	0.54	0.03	

(a) OnePlus in *meeting* environment

Raspberry Pi 4b				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	2167	0	0	2167
close	472	503	8	983
safe	30	307	9	346
***n*-pred**	2669	810	0	17
				accuracy: 0.77
				macro F_1-score: 0.50
precision	0.81	0.62	0.53	
recall	1.00	0.51	0.03	
F_1-score	0.90	0.56	0.05	

(b) Raspberry Pi 4b in *meeting* environment

Apple iPhone 6S				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	391	86	0	477
close	222	1606	223	2051
safe	29	234	147	410
***n*-pred**	642	1926	370	2938
				accuracy: 0.73
				macro F_1-score: 0.63
precision	0.61	0.83	0.40	
recall	0.82	0.78	0.36	
F_1-score	0.70	0.81	0.38	

(c) iPhone in *meeting* environment

OnePlus Nord N10 5G				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	9	29	0	38
close	1	47	34	82
safe	0	12	70	82
***n*-pred**	10	88	104	202
				accuracy: 0.62
				macro F_1-score: 0.56
precision	0.90	0.53	0.67	
recall	0.24	0.57	0.85	
F_1-score	0.38	0.55	0.75	

(d) OnePlus in *bus* environment

Raspberry Pi 4b				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	47	0	0	47
close	81	3	0	84
safe	87	0	0	87
***n*-pred**	215	3	0	218
				accuracy: 0.23
				macro F_1-score: 0.14
precision	0.22	1.00	0.00	
recall	1.00	0.04	0.00	
F_1-score	0.36	0.07	0.00	

(e) Raspberry in *bus* environment

iPhone 6S				
true/pred	**very close**	**close**	**safe**	***n*-true**
very close	50	1	0	51
close	68	39	3	110
safe	53	47	3	103
***n*-pred**	171	87	6	264
				accuracy: 0.35
				macro F_1-score: 0.30
precision	0.29	0.45	0.50	
recall	0.98	0.35	0.03	
F_1-score	0.45	0.40	0.06	

(f) iPhone in *bus* environment

Table 3.2: Performance metrics for BLE classifcation, thresholds on calibrated attenuation values

Table 3.2d shows the performance metrics for the OnePlus measured in the *bus* environment. From the 38 *very close* samples, only 10 have been predicted in the right class. This may result from the observation, that for the OnePlus, there was a higher base attenuation in the bus and *train* environment, than in the meeting set, that was used for calibration. The performance for the *close* class is slightly better than for the *very close* class. The best predicted class in this confusion matrix is the *safe* class. With an F_1-score of 0.75 it was well predicted. Overall for the OnePlus in the *bus* environment, the macro F_1-score is at 0.56. Now we continue with the results for the Raspberry, listed in table 3.2e. Almost all samples have been predicted to be in the *very close* class (215 of 218). This, of course, brings the recall to a high value for this class. But overall the classification result is very poor, with an f1-macro score of 0.14. Remembering back, to the overview plot in figure 3.11c, that is comparing the sets for a specific device. The *bus* environment showed much lower attenuation values than, the meeting set, for all distances. This is probably the reason that most of the samples were predicted as *very close* since the calibration was done using the *meeting* environment. Slightly better, but still a poor performance result can be retrieved for the iPhone in table 3.2f. Most of the samples were again mostly predicted as *very close* distances. It is not as bad as for the Raspberry, but is still only a precision of 0.45 in the *close* class. This all´ brings a macro F_1-score of only 0.3. This is probably also caused because the attenuations of the bus set were better than the meeting room set.

The last data set that needs to be presented is the train data set. The structure of these results is quite different, because only two distance classes have been measured, *very close* and *close* for the OnePlus and the iPhone, and only *close* for the Raspberry ($150cm$ and $160cm$). To get an overview of the quality of the predictions for this environment, we listed the f1-scores for the certain classes in table 3.3. For the OnePlus 180 of 493 samples have been correctly predicted in the *very close* class. The others were placed in the *close* class. This results in an F_1-score for *very close* of 0.54. The performance of the *close* class is worse, because of the wrong predicted samples that would belong into the *very close* class and a faulty prediction of the *close* samples themselves, where 237 of 288 landed in the *safe* prediction. Overall the performance was quite poor for the OnePlus classification. Next, the Raspbery Pi is also showing poor performance values, this is because 511 of the set size of 605 have been falsely predicted as *very close* instead of close. Only the iPhone performances in the *train* scenario were respectable, with a macro F_1-score of 0.7. Except for a bigger false prediction of 239 out of 515 *close* distance samples with a *very close* label, the set was predicted quite good. Even though this measurement was not measured on a measuring section, such as the bus and meeting set, we will include this data into the training set later. We are doing this, because the measured curves here behave differently to

	F_1-score		
class/device	**OnePlus N10**	**Pi 4b**	**iPhone 6S**
very close	0.53	NA	0.71
close	0.16	0.26	0.68
macro F_1-score	0.35	0.26	0.70

Table 3.3: f1-scores for BLE train data set, thresholds on calibrated attenuation values (Pi set only contains *close* range samples)

the other environments. By adding this data into the training set that is used for the machine learning approaches, we expect a better generalization of the model. Of course, some data will be kept out from this set, in order to use it for testing and evaluating the model. But this will be described in more detail in the end of this chapter.

We already compared our measured data against the empirical study of Nokoukar et al. [56], and we found out that our results in indoor settings were comparable to that study. Now with having the classification performance metrics we can make a comparison to the measurements that have been made by Leith and Farrell. In their last paper, they evaluated the potential of using BLE RSSI values to estimate the distance [45]. In this study, they showed several RSSI to distance plots with data measured in public transport scenarios. In these measurements, they had similar peaks and variance in the data. These results are quite similar to our measurements, of course it is varying on the device type. Moreover, in their papers on the accuracy of proximity tracing systems, for bus [48] and tram [47] environments, they presented classification results over time. Their classification was done on averaged attenuation values, as it was done before in GAEN. Their results showed a very poor performance, and almost no warnings were triggered over a time of 15 minutes. Compared to the false-positives rates, they presented, we have comparable results for certain device/environment combinations. For example, the iPhone or the Raspberry in the *bus* environment had, similar bad performances. But for the combinations where we presented f1-scores higher than 0.6, we got significantly better results. When we sampled the data over time, and ran a test based on the risk estimation formula used in the German proximity tracing application, we were able to achieve significantly better results. Our tests reported accuracy values (averaged over all distances) of 70% for the *meeting* environment. At certain distances, we observed quite bad performance, especially for 15 minute contact windows. By the weighted formula for risk estimation, this gets better for 30 minutes windows, when *close* distances are also taken into account by half. But overall, these results show that this procedure is not as bad, as it is stated by Leith and Farrell.

We assume that this significant improvement compared to the studies by Leith and Farrell resulted from using the minimum attenuation for the risk calculation (cf. section 2.11). This reduces the variance of the data significantly and no biased average over all three channels is used for risk calculation. However, when we classify the single RSSI values, as we did in our presented performance evaluation, we can see that there is still a lot of room for improvement, that may be achieved by combining different signal types. Next, we will have a look at some specifics of the data, we observed while working with the ground truth sets.

Data insights in the BLE data and resulting privacy issues

While working with the captured BLE packets, we found out that there was a significant difference. It turned out that the OnePlus Nord N10 5G we were using in the measurements does not send out the BLE payload as defined in the specification. In the packets sent, the flag section is missing. So the payload is directly starting with the 16-bit service UUID. To better illustrate this issue, two payloads, one from the OnePlus, and the other one send out by the iPhone, are presented in listing 3.1. As we can see in the listing, the OnePlus is directly starting with 336FFD as part of the service UUID and the iPhone is doing it as described in the specification, starting with the flag section. We investigated this issue with some more Android devices available in our team, and we found out that all 6 other devices had also this issue of sending to short payloads. By searching through the issues in the Github repository for the German proximity tracing app, we found an issue, that this might be an Android wide problem[11]. In the Github-issue, the developers of the application stated, that this issue should have been fixed with a newer version of the Android *exposure notification framework*. But after installing the newest version, we were still facing this issue. The same observation was made by member of the community on Github.

Listing 3.1: Comparing the GAEN BLE payload of an iOS device with an android device

```
# OnePlus Nord N10 5G
        3 3 6F FD 17 16 6F FD C4 52 CE CF 87 [...]
# Apple iPhone 6S
2 1 1A 3 3 6F FD 17 16 6F FD F0 E7 FD D6 9D [...]
```

[11] https://github.com/corona-warn-app/cwa-app-android/issues/782

This violation of the specification, done by Android devices, is very helpful for this thesis. Because it enables us to separate the number of MAC addresses, that may follow after an address and payload change, into two groups. A group for iOS devices that follow the specification, and a group of Android devices, that violate the specification. This reduces the number of possible next MAC addresses after a change and heavily increases the probability of identifying traces of MAC address changes. Moreover, because the market share of iOS devices is at 27% by the end of April 2021[12], the likelihood of identifying the next addresses for an iPhone is even higher. This specification violation problem can help in this work to develop approaches for better mapping of BLE and IEEE 802.11 signals, but one thing that is very worrying is that this violation of the standard raises a major privacy issue. Because it enables attackers, to have less noise in sniffed data and makes it easier to identify device types. For example, in a bigger shopping mall, it could be more likely to trace the position in a closed area that is equipped with BLE sniffers all around the area. By uncovering the MAC address and payload changes by a time and likelihood approach, it might be possible to track a device over the complete time it is in range of the receivers used for sniffing. This results in that an attacker is able to build a location graph for a given area, and also to link these positions or payloads that are sniffed with a COVID-19 positive status, when one GAEN user from this set publishes his or her infection status. When this position data is linked with a CCTV camera picture from the given area, it may be possible to get the identity of a person that was sending the BLE packets in the observed area. To solve this alarming issue Google needs to patch this problem, that the Android devices also follow the specification. This would higher the noise and sets all devices equal in a sniffed data set and with a high number k of devices in a trace it gets harder to uncover the MAC address traces.

Another issue, was found in the train data set, which also contains traces by other devices in the *close* area. At first in this set, we saw asynchronous MAC address and payload changes. This means, that the MAC address changed, but the payload containing the RPI stayed the same. After some time then, at the next MAC address, the device was able to change MAC address and payload at the same time. A pattern was easy to identify across the whole sniffed trace, and by having such a pattern a device like this can be re-identified, even without sniffing every MAC address change. One malicious trace (with a changed RPI, for anonymization purposes) is visualized in figure 3.12. This device was changing the MAC address but not the payload, sent two packets with the new MAC address, and then it changed the MAC address again, but this time synchronous with the RPI. Then, the device kept sending, until the pattern repeated in 10 to 15 minutes when the MAC address change was triggered again.

[12] `https://gs.statcounter.com/os-market-share/mobile/worldwide`

71:6a:55:45:b3:a8	3 3 6F FD 17 16 6F FD AB F3 A7 94 BD B 4C 44 17 B8 BC 73 41 85 28 74 45 AF CF 8A
56:26:ef:51:71:1b	3 3 6F FD 17 16 6F FD AB F3 A7 94 BD B 4C 44 17 B8 BC 73 41 85 28 74 45 AF CF 8A
56:26:ef:51:71:1b	3 3 6F FD 17 16 6F FD AB F3 A7 94 BD B 4C 44 17 B8 BC 73 41 85 28 74 45 AF CF 8A
7e:26:17:1b:04:ce	3 3 6F FD 17 16 6F FD 6B 6D 61 B3 5 1E 69 C6 9 FB AD E5 A6 38 D9 48 E5 73 24 F8

Figure 3.12: Malicious trace pattern observed by an Android device in the *train* environment

We assume that this was an older Android version running a legacy version of the exposure notification framework, with an issue in a synchronized change of RPI and MAC address. Such patterns were only observed for a few devices. For the devices used in the measurements, such an issue was not observed. In the next section, we will present the IEEE 802.11 ground truth data, collected in the same data sets as presented before.

3.6.2 IEEE 802.11 data

In this section, we will present the measured data for the IEEE 802.11 signals for the environments meeting, bus and train. Compared to the BLE data, presented before, we do not have any thresholds defined yet, that are used for proximity tracing. There is no base and no need for calibration. Thus, we will later build separate models for each device, and make the models device specific. Additionally, this means that we do not present attenuation values in this section, instead, we will look at the raw RSSI values measured on the sender side. The structure for the following: first, we will for each band inspect the data channel wise, in order to get information on the RSSI variance in a channel. Secondly, we will compare the three environments, for each device, to each other, based on the average value over all channels per distance. To get an overview of the propagation specifics, for each device in the different environments. Classification performance results, as we showed them for BLE, will not be presented in this section. This will be done later in the evaluation chapter, for different models applied to the data.

2.4 GHz band

For the 2.4 GHz band, the ground truth measurement is visualized as a box plot for each channel per distance in figure 3.13. At distances of $50cm$ and $100cm$, the very long box plots stand out directly. The observed RSSI range from -25 up to -65, for example for Frequency 2457. This is a very high variance. The frequencies where we can observe such high variances are mostly for a distance of less equal $100cm$. For higher distances, the boxes stay small with ranges of about $10dBm$ from minimum to maximum, on average even less. For all distances, even

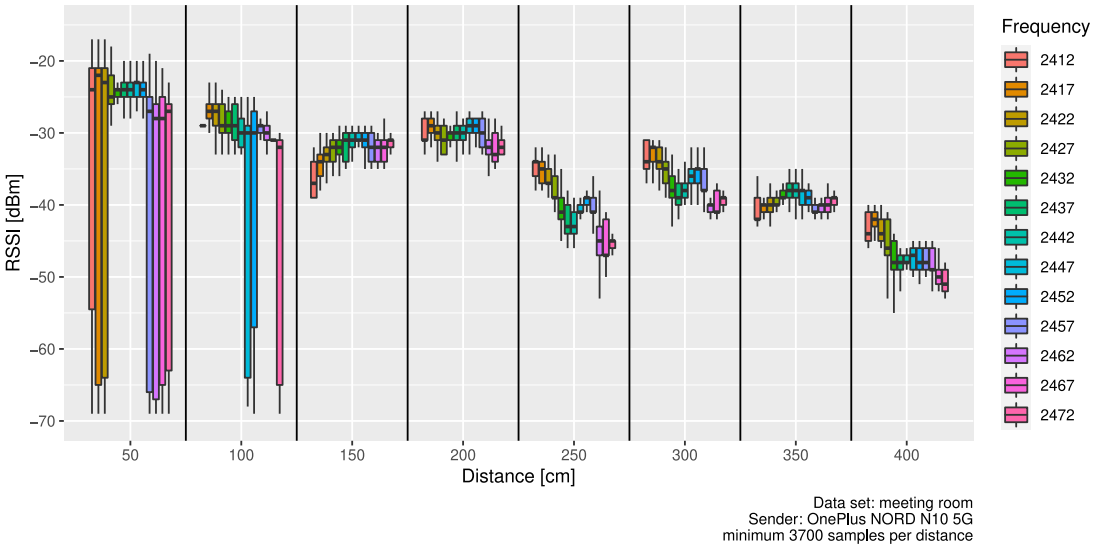


Figure 3.13: Comparison of the data measured for each channel on 2.4GHz IEEE 802.11

the small ones, the means are usually not far from each other across the different frequencies. But this only holds true for the OnePlus data, e.g., the distances of the means for the Raspberry Pi are higher, they can range up to $25dBm$ difference, in the same distance between, sender and receiver. However, for the other devices, we can also state, that the variance on distances of $50cm$ and $100cm$ is very high and gets minimal for higher distances. In figure 3.13 for the OnePlus, we can observe a worse signal strength at $250cm$, which results from the same issues that we also observed in the BLE data. At certain distances, the interference from the ground path is higher, because both signals reach the receiver at almost the same time. For this device and environment, it seems to be at $250cm$. Another observation we can make here is, that there is no pattern visible which channel have a better RSSI over the distance. For each distance, there is another channel, that has the higher average RSSI.

Next, we will compare the meeting set measured for the OnePlus with other sets measured with this sender. A plot showing the average RSSI values per distance is visualized in figure 3.14. As for the BLE plots, the points are only connected for having a better overview, this interpolation does not state a result that would actually be measured. Because it is likely that measurements in between the distances presented here, will show values that are not connected to this curve. Coming to the curves themselves, we can see long error bars, these are mostly caused because it is the standard deviation calculated across all measured frequencies. For

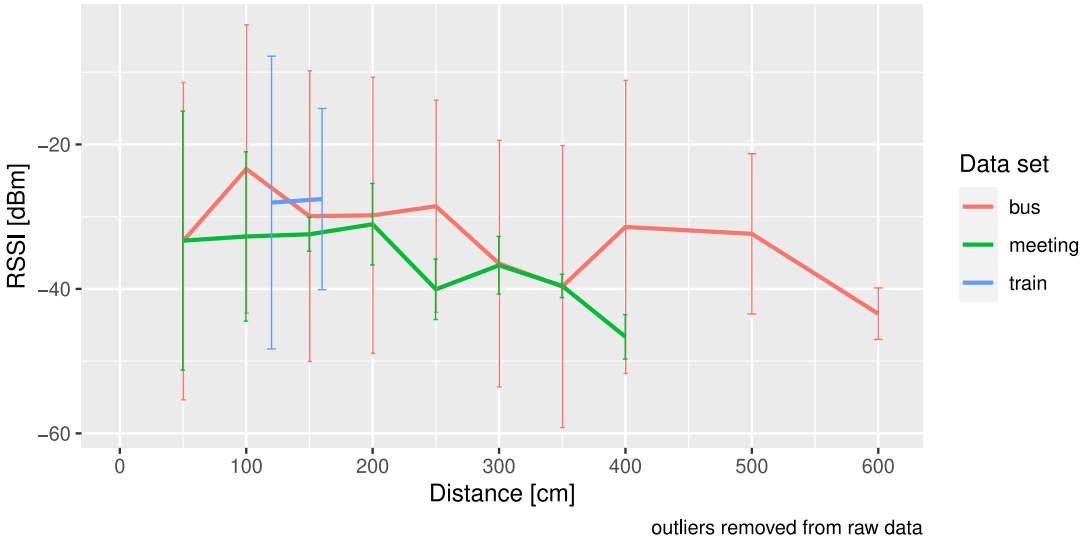


Figure 3.14: Comparison of the environments for the measured 2.4 GHz IEEE 802.11 signals (Sender: OnePlus Nord N10)

the train and bus set, the standard deviations are much higher compared to the meeting set. This is either caused by a high variance inside a certain frequency or high variances between the different frequencies. An investigation in the data measured in this environment showed that it is caused by both. There are certain frequencies that show box plots that are larger than $40dBm$, and also big differences comparing the average RSSI of certain channels in the same distance to each other. These differences between certain frequencies can be higher than $30dBm$. However, if we compare the average lines of these three environments to each other, we can state that the lines are much closer to each other, as it was the case for the BLE data measured for the OnePlus. The observations we made for the other devices using the 2.4 GHz band were comparable to the OnePlus. The graphs for the devices can be found in the appendix. Figure 6.1a represents the iPhone data and figure 6.1b the Raspberry Pi data.

To conclude this part of the ground truth analysis, the data measured at $50cm$ and $100cm$ has a high variance. It is likely that it will be challenging to make an accurate distance estimation for these distances using 2.4GHz IEEE 802.11 signals. But for higher distances the variance decreases, that is mostly around $5dBm$ and $10dBm$. However, this is in comparison to the minimum attenuation BLE data a bit higher. Nevertheless, due to the information which frequency was used for sending, we might be able to extract some patterns for certain frequencies at specific distances, which can help to estimate the proximity.

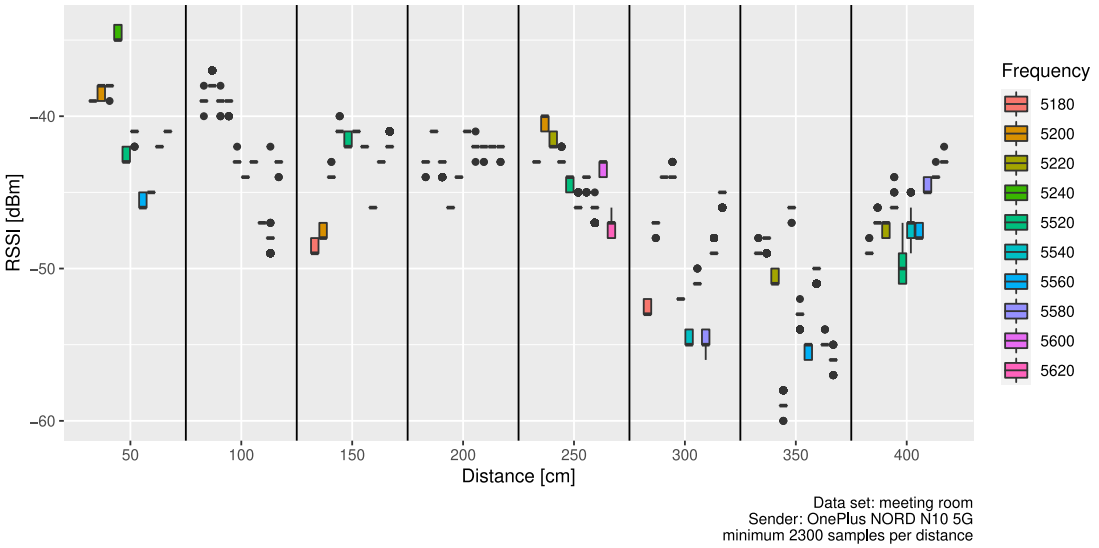


Figure 3.15: Comparison of the data measured for each channel on 5GHz IEEE 802.11

5 GHz band

Now, we will compare the observations made in the 2.4 GHz band to the data we captured for the 5 GHz band. This is likely to behave much differently, because unless the other two signal types we have seen before, it is not operating in the 2.4GHz band, and therefore it has other propagation characteristics.

The channel comparison for the OnePlus in the meeting room environment is visualized in figure 3.15. At first glance, you can see that there is much less variance within the channels compared to the 2.4 GHz band. For many channels, only the average line is drawn and no box plot was existent, because the RSSI values collected on a channel for the distance stayed constant. This is also the case for *very close* distances. Thus, the inner channel variance is quite low and the signal strength is constant, compared to 2.4 GHz. It is likely that the situation in 5 GHz is better due to less noise and other devices/technologies that are operating on this band. Another observation in this plot is, that the channels have different signal strength levels in the same distance. This was also observed in 2.4 GHz, in the two-ray ground-reflection model for the BLE frequencies. Each frequency has its' points where an overlay with the reflected ground signal is occurring. These differences in a range up to $12dBM$, are presenting such frequency specifics. Of course, it is also influenced by more paths than the ground path, and also by other factors that were presented in section 2.5. A two-ray ground-propagation model was also computed for the frequencies used in 5 GHz, this model is illustrated in

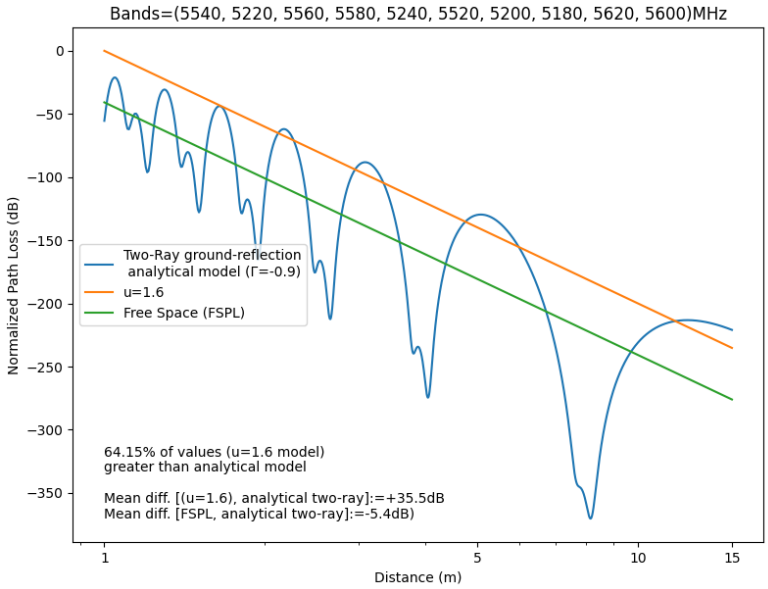


Figure 3.16: Analytical Two-Ray ground-reflection model for IEEE 802.11a bands channels for a height of 47cm (Calculated with model [66])

.

figure 3.16. For the other two devices, these plots, which illustrate the measured data per channel, look very similar in the meeting room environment. And except for some more variance in the very short and longer distances, also similar for the *bus* environment. These constant signal strengths for certain frequencies and resulting lower variances, are usually a better base for training machine learning models on such data.

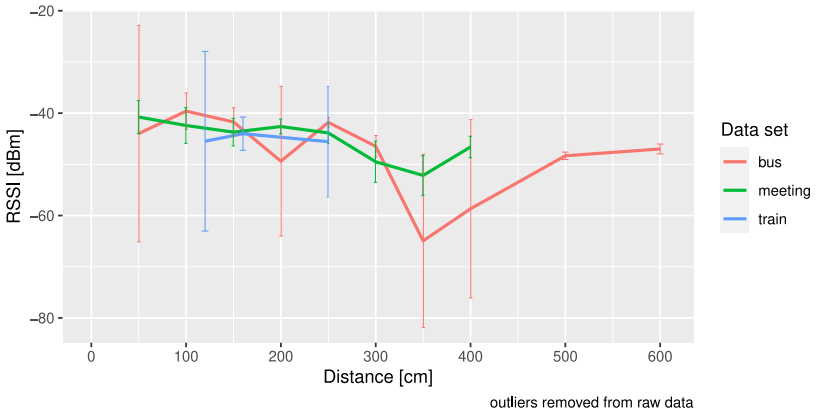
Next, we will compare the different environment, in the same way as we did it for the other two signal types. Starting with the OnePlus in figure 3.17a. It is interesting to observe that the average RSSI across all three environments is quite *close* to each other for the distance interval from $50cm$ to $400cm$, then the gap between meeting and *bus* environment is getting wider. In this plot, the high standard deviations, shown for the bus set, do mostly represent the variance of the channels to each other in a certain distance, and not the inner channel variance. Except for two outliers for frequency 5180 at $50cm$ and $200cm$ distance, there the inner channel variance was very high in the *bus* environment. The following plot in figure 3.17b illustrates the average RSSI for the iPhone 6S. In this plot, we have a very clear decrease with increasing distance up to $300cm$ for the *meeting* environment. Then a peak at $350cm$ stops this trend, it is also an optimum in the analytical model in figure 3.16. Such a decreasing trend for the RSSI with increasing distance is also observable for the *train* environment. For the bus it is

a less steady trend, but compared to the OnePlus plot, the standard deviation is lower. A similarity of the propagation observed for these two devices is an increase of the RSSI at higher distances in the bus set. Ending the comparison with the plot for the Raspberry Pi 4b in figure 3.17c. In the plot, we have a similar situation, as we saw it for the iPhone. A very nice and steady RSSI decrease over distance for the meeting room, with a peak at $300cm$, too. Moreover, for this device, we have a similar trend for the bus set. Both environments and also the *train* environment, show low standard deviation values. Except for the already mentioned issue at $600cm$ in the *bus* environment, there we face a high inner channel variance.

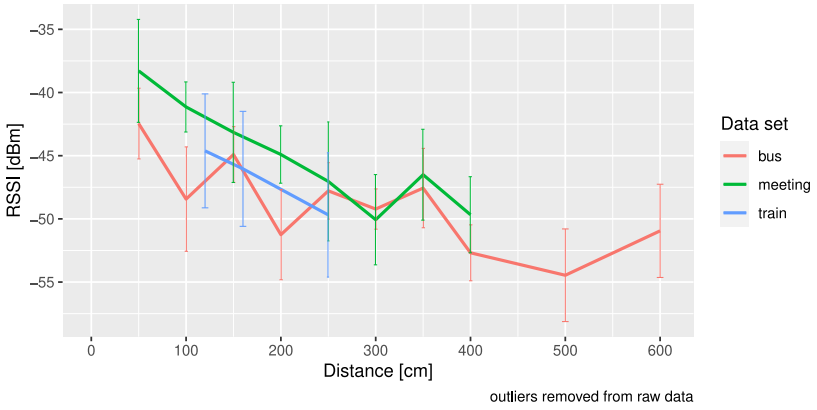
To conclude this assessment of the data captured on 5 GHz, it can be said that we have less variance in general for the measured RSSI values at certain distances. Moreover, by having average RSSI values per device that are much closer to each other, than this is the case for 2.4 GHz signals, we have a good data basis to perform a distance class estimation. Based on the observations of the previously discussed curves, we expect to achieve better performance results for 5 GHz data, than we have seen for BLE data. We will evaluate this in the next chapter, with various models. But first, we will end this chapter, by describing the concept of evaluation and a brief summary.

3.7 Summary

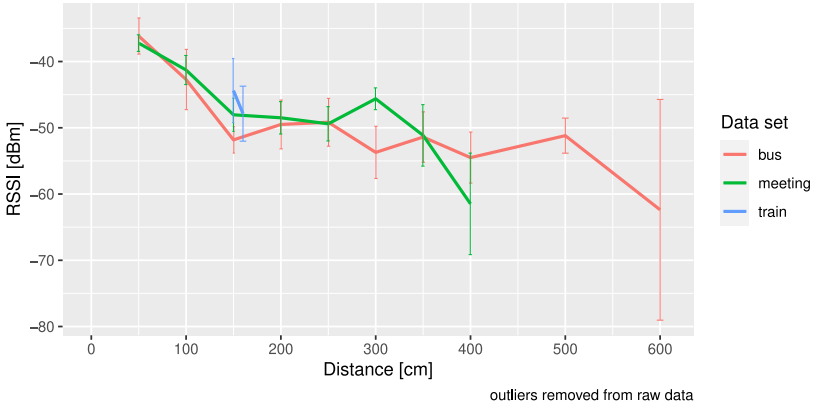
In this chapter, we presented approaches to tackle two issues, first on how we can improve the quality of proximity estimation by utilizing IEEE 802.11 probe requests, and second an approach on mapping these probe requests with the signals received via BLE. Then, we defined measurement goals, the setup and environments, to collect data, that can be used to implement and evaluate the approaches on. Next, we proposed a procedure for calibrating the BLE signals, that enables us to use the thresholds that are used for currently used proximity tracing solutions. This was followed by a presentation of the ground truth data collected for BLE and a report on the classification performance quality by using the GAEN approach. Afterwards we presented the data measured for IEEE 802.11 signals on 2.4 GHz and 5 GHz band, and we compared the quality of this data against the data collected using BLE. We concluded that this data, especially the one collected on 5 GHz band, looks very promising, in order to achieve a more accurate proximity estimation method.



(a) OnePlus Nord N10



(b) iPhone 6S



(c) Raspberry Pi 4b

Figure 3.17: Comparison of the environments for the measured 5 GHz IEEE 802.11 signals

4 Evaluation

In this chapter we will first present an approach that we will use for evaluation. Then we will evaluate the quality of a proximity estimation, by only using IEEE 802.11 signals. The signals will be evaluated separated by bands, 2.4 GHz and 5 GHz. These results will be compared to the *classic* BLE classification using thresholds. Then we will compare combined classifications using all three signal types. Moreover, the different models that are listed in the following will be compared to each other. Then we will evaluate these models with the help of other data sets. Next, we will discuss problems and limitations of these models. This is followed by an assessment of the mapping of IEEE 802.11 probe requests with BLE advertisements. Finally, we conclude with an assessment of the privacy implications such a mapping can bring.

4.1 Evaluation approach

To evaluate the performance of a proximity estimation approach utilizing BLE and IEEE 802.11 signals, the measured data needs to preprocessed and balanced. This is necessary in order to avoid training bias and over-fitting to certain environments or distances. In the following we will explain the data preprocessing steps. Then, we will define an order in which the approach and models will be evaluated, and after this we will show the metrics that will be used for evaluation. In the end, we will explain the plan for evaluating the packet matching for IEEE 802.11 and BLE signals.

Data preprocessing

In order to balance the data-sets of different sizes we have to up- or downsample the data. On the one hand, by down-sampling all the data to the smallest data sets, a lot of information will get lost, from the bigger data-sets. In the worst case measured data for certain distances will not be included, by using a random down-sampling approach. On the other hand, by performing upsampling, certain measurement errors may be inflated. But since we inspected most of the data,

and because of the analysis over time, where we showed that on average the collected data is stays constant, we assume that these problems that may result from upsampling are negligible and are significantly less problematic than the loss of information that will result from down-sampling. To perform the upsampling we merged all the data-sets per device together and iterated over each distance in each scenario. Then we upsampled the data of the distance to a sample size of $50,000$, this is done for each distance of an environment and for each signal type. In the merged data set we now have $50,000$ samples per distance and per environment. We chose this sample size to obtain sets of minimum $100,000$ for each class. This data set is by factor 10 larger than our largest measured class in the data sets, by this we want to avoid over representation of larger measured sets. Then the three signal types were merged into one table. Resulting in set that contains $50,000$ samples for each signal type (BLE, IEEE 802.11 2.4 GHz and 5 GHz), for each distance. In the next step for each device type and distance class, *very close*, close, and safe, we will take a random sample of $100,000$ samples. In this process we made sure, that each distance is equally represented in the data set. This then, for example results in a random sample for the *very close* class, that contains $50,000$ samples for $50cm$ and the same number for $100cm$. These $100,000$ samples taken for each class will be then combined by merging them together. It results in a table of $300,000$ rows, containing the attenuation for a BLE signal, RSSI for the 2.4 GHz and 5 GHz IEEE 802.11 signals, and the specific frequencies in the bands, for each row.

This overall set was then split into a training, test and evaluation set. The training set consists of 60% of the samples, and the test and evaluation set of 20% each (c.f. [55]). The split random samples were taken by stratifying over the class. This means that we have the same distribution of classes in each set. For implementing and testing the models, the training and test set were used. The evaluation set was saved and only used later for a final evaluation. Moreover, we have two other evaluation sets, that were mentioned in the *meeting* and *bus* environment. These data is combined in the same way, as it was done for the other data sets, just with a smaller sample size. The complete data flow described in this text is shown in figure 4.1 for a better overview.

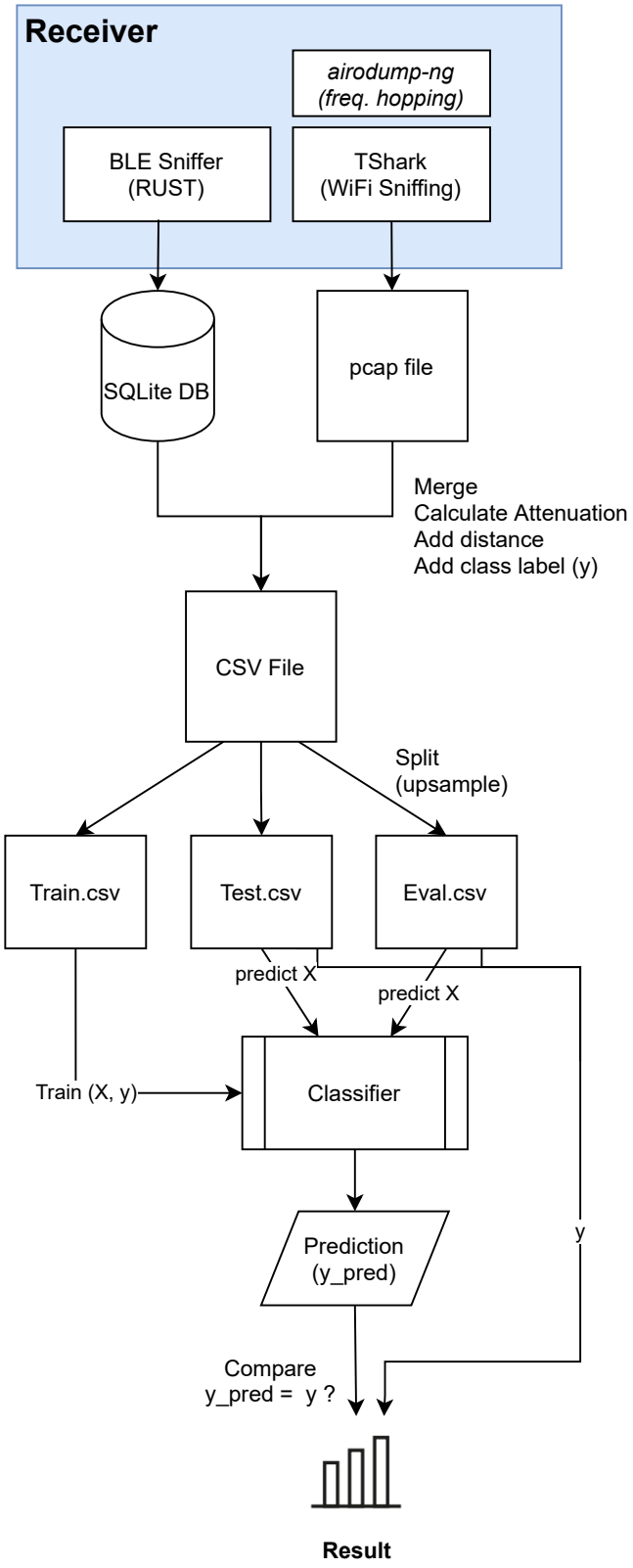


Figure 4.1: Data-flow diagram showing the data processing from capturing to the evaluation result

Models implemented and evaluation steps

The following models will be implemented and evaluated:

1. IEEE 2.4 GHz decision tree
2. IEEE 2.4 GHz decision random forest
3. IEEE 5 GHz decision tree
4. IEEE 5 GHz decision random forest
5. Combined specialized classifiers with decision trees (unweighted)
6. Combined specialized classifiers with random forests (unweighted)
7. Combined specialized classifiers with decision trees (weighted)
8. Combined specialized classifiers with random forests (weighted)
9. Combined general classifier decision tree
10. Combined general classifier random forest

At the beginning, we will evaluate the IEEE 802.11 single-channel classifier to get an overview of how well the classification for these signals individually compares to BLE. The specialized combined classifiers will be implemented, either with a random forest model or with a decision tree model is used for the IEEE 802.11 data measured, and the *classic* threshold approach is used for the BLE data. For the weighted combined classifier the IEEE 802.11 only classifier that showed the best rating will get a higher weight, the weights will be presented in the evaluation chapter.

Evaluation metrics

For evaluating the implemented models, we will use the evaluation data-sets, as described before. Based on these sets, classification performance metrics will be calculated. These are the same metrics that have already been used for assessing the quality of the BLE ground-truth data. These are standard metrics for machine learning evaluations, by doing so we can assess the overall performance of a model and in detail the performance for certain classes, in order to identify difficulties for certain distance classes. This list of metrics will be computed:

- Confusion matrix
- Precision and recall for each class
- F_1-score for each class and for the overall set/model

- Accuracy for the overall set/model, which is now meaningful on the basis of this data, because the classes are balanced now

Using these metrics, we will be able to illustrate the performance of the implemented machine learning models. Moreover, we will compare these results to the results that were achieved by using BLE only.

Evaluation of IEEE 802.11 and BLE signal matching

The second issue of matching IEEE 802.11 and BLE signals will be evaluated and analysed manually. Using the approach of identifying sequences of MAC addresses used by a device over time (c.f. section 3.1.2). Resulting in sequences of MAC addresses for each signal type, BLE and IEEE 802.11. We will compare the sequences found by our approach, with our manually extracted sequences, in order to assess the quality. Then we will discuss the limitations of such an approach, and we will provide an analysis of privacy implications that result from having the probe requests mapped to the BLE signals. Moreover, we will give a short analysis on the approach of performing a direct matching based on RSSI values.

4.2 Set specific BLE results

To have base for comparisons, we have listed the classification results for BLE only with the OnePlus used for sending, using Thresholds based on the minimum attenuation values, in table 4.1. Data that was recorded for the other devices will be presented in a later following overview table of all used classifiers.

The data for the OnePlus shows, that almost half of the *very close* samples were falsely classified into the *close* class. The same occurred for the *safe* class, about $8,000$ samples were assigned into the *close* class, which drives the precision of this class down. Overall, the accuracy and F_1-score of the BLE classification are quite poor with values less than 0.6. Compared to the metrics, that were directly calculated on the sets measured for the OnePlus, with this result for the upsampled set, we see that these results are better. For the *train* and *meeting* set, we had f1-scores of 0.3, 0.35, and 0.56 in the *bus* set. The metrics of the upsampled set now, are better compared to the classification on the raw data that we presented in section 3.6. This is probably a result of upsampling, which reduced the extreme points present in the raw data.

OnePlus Nord N10 5G				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	10735	9148	117	20000
close	1921	11720	6359	20000
safe	325	8083	11592	20000
n-**pred**	12981	28951	18068	60000
				accuracy: 0.57
				macro F_1-score: 0.58
precision	0.83	0.40	0.64	
recall	0.54	0.59	0.58	
F_1-score	0.65	0.48	0.61	

Table 4.1: BLE classification for the OnePlus sender on the combined data evaluation set

4.3 IEEE 802.11 2.4 GHz evaluation

In this section we examine the classification quality when we are only using IEEE 802.11 2.4 GHz data. Since we do not have proximity tracing thresholds for this signal type, we will use a decision tree as described earlier. The model is trained with the 60% training sub-set, using the 2.4 GHz RSSI and the specific sending frequency in the 2.4GHz band as training features. A limit for the decision tree's depth is set at eight. This limit was chosen after empirical tests. By iterating and testing the tree for each maximum depth level, we found that we will reach a local performance maximum by the use of eight layers. With higher numbers the results only show slight improvement. We investigated this further by plotting the tree, and with higher maximum depth the splits are getting very small. Less than 100 samples are then split in the lower layers. Moreover, these splits do not result from rationality, like splitting for a very high attenuation into the *very close* class in a certain branch. This then, is an over-fitting to the scenarios we recorded. We will also use this maximum tree depth for random forests, that will be evaluated later.

The actual performance metrics for 2.4 GHz probe requests are listed in table 4.2. We listed the results for all three devices, starting with the OnePlus again, in table 4.2a. Both classes, *very close* and *safe*, show an F_1-score of 0.79. For the *close* class it is 0.65, almost one fourth of the samples that would belong to this class have been predicted as *very close*. In terms of proximity tracing and alarming people, 83% of the *very close* samples have been predicted correctly. This is almost 30% more than using BLE for classification. Overall we have a macro F_1-score of 0.74 for the IEEE 802.11 2.4 GHz samples. This is a plus of 0.16 compared to the BLE performance.

OnePlus Nord N10 5G				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	16538	2573	889	20000
close	4572	12610	2818	20000
safe	824	3598	15578	20000
n-**pred**	21934	18781	19285	60000
			accuracy: 0.75	
			macro F_1-score: 0.74	
precision	0.75	0.67	0.81	
recall	0.83	0.63	0.78	
F_1-score	0.79	0.65	0.79	

(a) 2.4 GHz OnePlus Nord N10

Apple iPhone 6S				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	16391	1277	2332	20000
close	1755	12123	6122	20000
safe	616	2206	17178	20000
n-**pred**	18762	15606	25632	60000
			accuracy: 0.76	
			macro F_1-score: 0.76	
precision	0.87	0.78	0.67	
recall	0.82	0.61	0.86	
F_1-score	0.85	0.68	0.75	

(b) 2.4 GHz Apple iPhone 6S

Raspberry Pi 4b				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	14930	2381	2689	20000
close	2866	13403	3731	20000
safe	2668	3229	14103	20000
n-**pred**	20464	19013	20523	60000
			accuracy: 0.71	
			macro F_1-score: 0.71	
precision	0.73	0.70	0.69	
recall	0.75	0.67	0.71	
F_1-score	0.74	0.69	0.70	

(c) 2.4 GHz Raspberry Pi 4b

OnePlus Nord N10 5G				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	16711	3266	23	20000
close	662	17149	2189	20000
safe	2	3232	16766	20000
n-**pred**	17375	23647	18978	60000
			accuracy: 0.84	
			macro F_1-score: 0.85	
precision	0.96	0.73	0.88	
recall	0.84	0.86	0.84	
F_1-score	0.89	0.79	0.86	

(d) 5 GHz OnePlus Nord N10

Apple iPhone 6S				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	15966	3079	955	20000
close	2788	13719	3493	20000
safe	1513	2573	15914	20000
n-**pred**	20267	19371	20362	60000
			accuracy: 0.76	
			macro F_1-score: 0.76	
precision	0.79	0.71	0.79	
recall	0.80	0.69	0.80	
F_1-score	0.79	0.70	0.79	

(e) 5 GHz Apple iPhone 6S

Raspberry Pi 4b				
true/pred	**very close**	**close**	**safe**	n-**true**
very close	19242	23	735	20000
close	1020	15082	3989	20000
safe	139	2590	17271	20000
n-**pred**	20401	17695	21904	60000
			accuracy: 0.86	
			macro F_1-score: 0.86	
precision	0.94	0.85	0.79	
recall	0.96	0.75	0.86	
F_1-score	0.95	0.80	0.82	

(f) 5 GHz Raspberry Pi 4b

Table 4.2: Performance metrics for decision trees on the IEEE 802.11 ground truth evaluation set.

Next, the performance metrics for the decision tree trained and evaluated with sending data from the iPhone in table 4.2b. The results are quite comparable to those from the OnePlus. We have a better precision for the *very close* class, because fewer samples from the *close* class have been misclassified. More than 2,000 samples from *very close* are false-negatives, they were predicted as *safe*. The class with the worst results is, for the iPhone also, the *close* class with an F_1-score of 0.68. It mostly results from the recall at 0.61, more than 6000 samples were false predicted as safe, instead of close.

Finally, we look at the third device, the Raspberry Pi 4b. The evaluation results are listed in table 4.2c. Compared to the other devices, the Pi has the worst F_1-score with 0.71 for the 2.4 GHz probe request classification. This results from weaker results in the *very close* and *safe* class. Here we have lower precision and recall values, compared to the other two devices. But with an overall F_1-score of 0.71 it is still much better than the BLE only results for the Raspberry Pi 4b, which is at 0.29 (see table 4.2c).

In conclusion, regardless of the device type, we can obtain more accurate classification results with the decision tree used with 2.4 GHz IEEE 802.11 probe requests than with the BLE threshold approach. Even at *very close* distances, where we had high variances in the measured data, the recall results, with higher than minimum 0.75, are quite good and unexpected. This improvement in comparison to the BLE approach probably results from the usage of a decision tree itself, because in a decision tree with the depth of 8 layers, there are more fine-grained splits than just the splitting by three thresholds. Secondly, the results are improved by using a second feature, namely the frequency on which the packet was received. Only the addition of the second feature allows us to make an informed partitioning with the decision tree. Otherwise, with only one feature, RSSI or attenuation, we will basically over-fit a decision tree onto the measured values, that can significantly change when the environment changes. Even with the frequency feature, the information level is very low, as we will discuss later in this chapter.

4.4 IEEE 802.11 5 GHz evaluation

In table 4.2 we also listed the performance metrics achieved for a decision tree classification using IEEE 802.11 probe requests that were captured on the 5 GHz band. For the decision tree we used a maximum depth of 8 layers, just like we did for 2.4 GHz. The tree was then trained using RSSI and frequency for each probe request packet captured. For training, testing and evaluation we used our upsampled combined data set again.

We start with the OnePlus results in table 4.2d. With this device we obtained a macro-f1 score of 0.85, this is 0.11 better than the 2.4 GHz result. Issues we had in classifying the *close* distances in 2.4 GHz (recall: 0.63), are less prominent (recall with 5GHz: 0.86). This results in better precision values for *very close* and *safe* distances. That again, then results in a better overall result using this band. Especially in the *very close* class, we achieved an F_1-score of 0.89, which is the most important class in order to assess the risks of an infection. The false-negatives for the *very close* samples, were mostly predicted as *close* distances (3266 samples) and only 23 as safe. The false-negatives in the *safe* class are the most severe, because they do not count in the risk assessment. But as in 2.4 GHz band, we also have more than 3000 samples that originally belong to the *safe* class, that have been predicted as *close* distance. The observations are consistent to our interpretation of the measured data, where we saw smaller variances in the data on *very close* distances.

Below, in table 4.2e, the performance evaluation for the iPhone is listed. With a macro F_1-score of 0.76 it is equal to the 2.4 GHz's evaluation. However, comparing the results class wise, we can observe a slightly better classification for *close* and *safe* distances. With an equal decrease in the quality of the *very close* prediction.

Finally, the Raspberry Pi 4b results on 5 GHz data listed in table 4.2f. Comparable to the OnePlus, this model also showed better results on the 5 GHz band than on the 2.4 GHz band. In fact, it did so well that the f1 score for each class was at least 0.8. This result is quite impressive compared to the poor result we observed for the Raspberry Pi using BLE only.

Concluding, the 5 GHz result we can see even more improvement, compared to the 2.4 GHz results, that were already better than using the BLE approach. On average the F_1-score for all three devices on 2.4 GHz was at 0.74, for 5 GHz now it is at 0.82. For certain distances for the iPhone, for example, we saw that 2.4 GHz was performing better on the *very close* class. Then for *close* and *safe* distance 5 GHz was doing better. Combining these together, as it is the approach we want to follow, will probably result in an even better classification. In the next section we will first evaluate, whether we can achieve better results for the specialized classifiers using random forest models. Afterwards, we will evaluate the combination of the three signal types.

4.5 Comparing decision tree and random forest as specialized classifiers

The same evaluation we did for the specialized decision tree classifiers before, was also performed using a random forest model. First tries without a fixed tree size, resulted in f1-scores of 1.0 for our evaluation set, which resulted from huge trees that were over-fitting. To get a comparable situation and no over-fitting we fixed the maximum depth for the trees in the random forest to eight, too. With these settings, we used the same training, testing and evaluation data from our upsampled data set. Compared to the 2.4 GHz decision tree, the random forests for each device perform as follows:

- Random forest 2.4 GHz classifier (macro f1-scores)
  - OnePlus Nord N10 5G: 0.75 (DT: 0.74)
  - Apple iPhone 6S: 0.77 (DT: 0.76)
  - Raspberry Pi 4b: 0.75 (DT: 0.71)
- Random forest 5 GHz classifier (f1-scores)
  - OnePlus Nord N10 5G: 0.87 (DT: 0.85)
  - Apple iPhone 6S: 0.79 (DT: 0.76)
  - Raspberry Pi 4b: 0.87 (DT: 0.86)

For the 2.4 GHz band we can observe slightly better f1-scores for the OnePlus and the iPhone, with an improvement of 0.01 and 0.02. The performance improvement or the Raspberry accounts 0.04. Similar results were retrieved for the classification on 5 GHz data. Here the OnePlus improved by 0.02, the iPhone by 0.03 and the Raspberry by 0.01. These slight improvements in both bands are probably a result of the ensemble learning method, by having various trees that are trained on a randomly selected local set, these trees profit then from each other by combining their results. In further evaluations we will provide results, for both, a decision tree and a random forest, that have been combined to a new multi-channel classifier. In general, we consider that the benefits of the random forest are not yet evident in our data. As more data is measured in other environments and used for training, we expect the benefits of random forest to become more pronounced.

4.6 Evaluation of combined specialized classifiers

We combined the classifiers presented before for 2.4 GHz and 5 GHz IEEE 802.11 signals and the BLE threshold based classifier together. For each data sample we put the signal type specific data, for example, 5 GHz RSSI and frequency, into the specialized classifier to retrieve a prediction. Then we combined these predictions into an overall result. In first place we translated our classes into numbers, this means the triple (*very close*, *close*, *safe*) will be represented by $(0, 1, 2)$. Each number in this triple is the result of one specialized classifier. Then, the numbers will be summed up and divided by three. In our example the result would be 1, which means that this sample is predicted as *close*. This is the unweighted approach. Moreover, we computed a weighted approach, where we multiplied each specialized result with a weight factor, before summing them up. As described, we evaluated random forest and decision trees in this approach. Next, we will present the results, starting with the unweighted approach.

4.6.1 Unweighted combination

For this and the following evaluations, we created a table that contains all classifiers that were evaluated in this thesis per device. The results are listed in table 4.3. In this overview, we present the F_1-score per class, the macro F_1-score, and the accuracy values. We are going to discuss the results for both, decision tree and random forest, combined classifiers, per device now.

We start with the OnePlus in table 4.3a. For the combined classifier using decision trees, the macro F_1-score is at 0.85. The results for the random forest based classifier is slightly better with 0.86. This improvement is comparable to the improvements we saw for the single channel classifiers, when we compared the decision tree performance to those of the random forest. The combined classifier is almost on an equal performance level as the 5 GHz single channel decision tree. The overall accuracy for the combined classifier is 0.02 less than the single decision tree. For the combined trees we see a little improvement by 0.1 in classifying the *very close* distances, but also a decrease in the *safe* distances. This is the result of one of the other two classifiers having the same prediction as the 5 GHz classifier, resulting in almost equal performance metrics. The same observation also holds for the combination of random forests. Additionally, the random forest combination also suffers a bit in classifying the *close* distances by 0.03.

A different observation, can be made for the unweighted combined classifier for the iPhone in table 4.3b. For both underlying models, the combined classification is worse than a single channel prediction for IEEE 802.11 signals. The classification

for the samples of *very close* distances is quite good with both models having an F_1-score of 0.85. This is an equal performance to the 2.4 GHz single channel decision tree. However, for the *close* and *safe* class, the prediction quality becomes worse than that of the single-channel decision tree in the 2.4 GHz band. This does actually result from the very bad BLE result in these classes (f1 0.2 in *safe* class). This lowers the chance of a correct match with one of the other classifiers, which are also not as accurate as those from the OnePlus. But still the IEEE 802.11 classifiers hold the f1 score up, above 0.6, and it is far better than we observed for BLE thresholds.

The situation is even worse for the unweighted classifiers used with the Raspberry Pi data, in table 4.3c. With macro f1-scores of 0.54 using decision trees and 0.55 using random forests, the combined classification suffers heavily from the bad BLE results. This impact is most significant in the *safe* class. The BLE F_1-score for this class is at 0.01. By the combination it gets up to 0.18, but this is still an unacceptable classification performance.

Overall the unweighted combined classification approach (with RF) has an average F_1-score of 0.71, which is worse than, for example just using the 5 GHz single channel classifier, that has an average F_1-score of 0.84. The key problem we faced, are inaccurate BLE predictions that count in by one third of the combined result. This issue may be solved by the weighted approach now.

4.6.2 Weighted combination

In the weighted approach, we first carried out an investigation using the testing data in order to obtain optimal weight factors. The approach here is, by knowing from our previous measurements, that the combined classification suffers from bad BLE predictions, reducing the weight of the BLE predictions. Moreover, since two of our three signal types operate on the 2.4 GHz band, we could also use a weighting that takes the two bands into account. In the combinations, we tested we also excluded the BLE data completely by setting the weight to zero. But it turned out that for *very close* distances the BLE prediction is quite helpful in improving the result. Eventually, we came up with the following weight formula after testing several combinations:

$$1.5 * 5\text{GHz}_{\text{pred}} + 0.75 * 2.4\text{GHz}_{\text{pred}} + 0.75 * \text{BLE}_{\text{pred}} \tag{4.1}$$

This weighting ensures that the predictions from both bands will be accounted equally in the final result. Additionally, by having a balanced weighting for the two 2.4 GHz technologies, they complement each other.

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy**	**macro** F_1-**score**
BLE thresholds only	0.65	0.48	0.61	0.57	0.58
IEEE 2.4 GHz decision tree	0.79	0.65	0.79	0.75	0.74
IEEE 2.4 GHz random forest	0.79	0.65	0.82	0.76	0.75
IEEE 5 GHz decision tree	0.88	0.79	0.89	0.86	0.85
IEEE 5 GHz random forest	0.89	0.83	0.91	0.88	0.87
Unweighted combi special DT	0.89	0.79	0.86	0.84	0.85
Weighted combi special DT	0.91	0.79	0.89	0.87	0.86
Unweighted combi special RF	0.91	0.80	0.87	0.86	0.86
Weighted Combi special RF	0.92	0.82	0.91	0.88	0.88
Combi general DT	0.94	0.88	0.94	0.92	0.92
Combi general RF	0.95	0.92	0.97	0.95	0.95

(a) Sender: OnePlus Nord N10 5G

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy**	**macro** F_1-**score**
BLE thresholds only	0.68	0.42	0.20	0.50	0.43
IEEE 2.4 GHz decision tree	0.85	0.68	0.75	0.76	0.76
IEEE 2.4 GHz random forest	0.85	0.71	0.75	0.77	0.77
IEEE 5 GHz decision tree	0.79	0.70	0.79	0.76	0.76
IEEE 5 GHz random forest	0.79	0.74	0.83	0.79	0.79
Unweighted combi special DT	0.85	0.65	0.60	0.70	0.70
Weighted combi special DT	0.87	0.71	0.81	0.80	0.79
Unweighted combi special RF	0.85	0.67	0.62	0.71	0.71
Weighted combi special RF	0.87	0.75	0.84	0.82	0.82
Combi general DT	0.95	0.83	0.86	0.88	0.88
Combi general RF	0.98	0.88	0.90	0.92	0.92

(b) Sender: Apple iPhone 6S

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy**	**macro** F_1-**score**
BLE thresholds only	0.56	0.32	0.01	0.41	0.29
IEEE 2.4 GHz decision tree	0.74	0.69	0.70	0.71	0.71
IEEE 2.4 GHz random forest	0.80	0.72	0.74	0.75	0.75
IEEE 5 GHz decision tree	0.95	0.80	0.82	0.86	0.86
IEEE 5 GHz random forest	0.95	0.80	0.84	0.87	0.87
Unweighted combi special DT	0.85	0.60	0.18	0.60	0.54
Weighted combi special DT	0.91	0.75	0.78	0.82	0.81
Unweighted combi special RF	0.88	0.60	0.17	0.62	0.55
Weighted combi special RF	0.91	0.77	0.80	0.83	0.83
Combi general DT	0.96	0.85	0.86	0.89	0.89
Combi general RF	0.98	0.89	0.90	0.92	0.92

(c) Sender: Raspberry Pi 4b

Table 4.3: Classifier overview for the ground truth evaluation set

Results of our evaluation confirm the quality of the weighting factors we have chosen. The metrics for the One Plus in table 4.3a, using a decision tree in the combination, we can observe slightly better results than the best single channel (decision tree) classifier, the 5 GHz classifier. The improvements have been mainly in the prediction of the *very close* class. The F_1-score improved to 0.91. A similar observation can be made by inspecting the random forest results. With the iPhone data, in table 4.3b, the weighted approach also resulted in an improvement in predicting the *safe* class. Only the Raspberry Pi seems to be an outlier in these observations. For this device the 5 GHz single-channel classifier showed better performance results. But the weighted combined classifier, improved a lot compared to the unweighted one. Even with a lower weight, the very weak prediction results from the BLE classifier push the combined result down. In section 4.8, we are going to evaluate these combined classifiers, and the genera classifier presented next, against additional evaluation sets, that are unseen by the models so far.

4.7 Evaluation of a general classifier

The last classifiers not evaluated yet are the general classifiers. Again the base is a decision tree or random forest. Instead of training the models using only the data of a single channel, however, we will use all channels now. This will result in a decision tree that uses the features that result in the biggest splits. Subtleties, such as more accurate splitting of a certain distance class using a signal type that performs good in this class, will automatically apply during the training of the tree. Moreover, it results in more fine-grained splits than just setting up weight factors that apply to all classes. The results are listed in table 4.3 again. Regardless of the device type, we can see that the general classifier outperforms all other classifiers, we have evaluated before. By using a decision tree, the results are *close* or above a macro F_1-score of 0.90. With a random forest, all devices were above 0.90. The best results can be found for the iPhone and Raspberry Pi, where the f1-scores in the *very close* class achieved a result of 0.98.

Concluding, after we inspected the various types of classifiers, we can state that the best result was achieved with a general classifier, that is able to utilize all features for training. Such a classifier uses the signal type that achieves the purest split to distinguish between classes. Or in other words, the signal type that brings the most knowledge for splitting the current class is used for splitting. At first, we suspected that mainly the strong features, such as the IEEE 802.11 signals were used for splitting. However, after we plotted and checked the trees, we found that the features were used in a very varied way.

4.8 Additional evaluations

In order to assess the quality of our approach on unseen and not trained scenarios and situations, as they will also occur in daily-use of a proximity tracing system, we kept the seat scenarios settings from the *bus* environment and the *meeting* scenario settings out so far. These two sets will now be used to evaluate the already trained models.

4.8.1 Bus scenario evaluation set

The *bus* scenario contains several iterations, where we placed devices on different seats in the *bus*, to model a real-life *bus* driving scenario. Details of the data set are described in section 3.4.3. As we did it in the previous data sets, for BLE we extracted the minimum attenuation values per time window. Then we matched IEEE 802.11 signals to that BLE data. For each band, we used the sample with the smallest time delta (previously filtered by device type and distance), to the BLE sample. This resulted in a data set with 182 samples. This set is smaller, than those seen before, because we captured each scenario for half an hour and then only one third of the data was used, due to the minimum attenuation. The classes in this set are not balanced, resulting in biased accuracy values. We let all previously presented classifiers predict on the attenuation, RSSI and frequency data. The results are listed in table 4.4.

For each device in this table, the BLE classification performance is in between a macro F_1-score from 0.2 and 0.3. Moreover, the 5 GHz single channel classifiers perform better than the 2.4 GHz single channel classifiers, with f1-scores in 5 GHz from 0.55 to 0.65. The combined specialized classifiers show similar results as the 5 GHz single channel classifier. The weighted approach shows better results than the unweighted approach, except for the Raspberry, there it is vice-versa. Finally, the best performing models on this data set are the two general classifiers, with macro f1-scores higher than 0.7 up to 0.8 in some cases. Interestingly, for the iPhone and OnePlus the decision tree outperforms the random forest with a small margin. For the Raspberry data the random forest performs better. This difference results probably from random samples that contain more observations from the *bus* ground-truth data, used for training. In conclusion our results obtained by the general classifiers are more than two times better than the results obtained by using the BLE only approach. Additionally, the evaluation of this *bus* scenario set showed that the good results presented before on the upsampled data set do not result from over-fitting and these models are generalized to work also on other data.

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds	0.00	0.02	0.67	0.37	0.23
IEEE 2.4 GHz decision tree	0.69	0.37	0.43	0.49	0.50
IEEE 2.4 GHz random forest	0.65	0.37	0.47	0.50	0.50
IEEE 5 GHz decision tree	0.69	0.55	0.65	0.62	0.63
IEEE 5 GHz random forest	0.68	0.58	0.68	0.64	0.65
Unweighted combi special DT	0.63	0.50	0.65	0.58	0.59
Weighted combi special DT	0.71	0.48	0.68	0.62	0.62
Unweighted combi special RF	0.61	0.51	0.67	0.59	0.60
Weighted combi special RF	0.71	0.53	0.71	0.64	0.65
Combi general DT	0.83	0.74	0.83	0.79	0.80
Combi general RF	0.76	0.68	0.74	0.72	0.73

(a) Sender: OnePlus Nord N10 5G

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds	0.66	0.00	0.15	0.35	0.27
IEEE 2.4 GHz decision tree	0.38	0.41	0.46	0.42	0.42
IEEE 2.4 GHz random forest	0.30	0.50	0.46	0.44	0.42
IEEE 5 GHz decision tree	0.68	0.42	0.62	0.58	0.57
IEEE 5 GHz random forest	0.69	0.50	0.49	0.58	0.56
Unweighted combi special DT	0.46	0.43	0.57	0.47	0.49
Weighted combi special DT	0.69	0.37	0.56	0.55	0.54
Unweighted combi special RF	0.38	0.43	0.47	0.42	0.43
Weighted combi special RF	0.71	0.48	0.53	0.59	0.58
Combi general DT	0.83	0.63	0.78	0.75	0.75
Combi general RF	0.85	0.62	0.75	0.74	0.74

(b) Sender: Apple iPhone 6S

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds only	0.47	0.02	*N/A*	0.31	0.25
IEEE 2.4 GHz decision tree	0.46	0.26	*N/A*	0.29	0.36
IEEE 2.4 GHz random forest	0.38	0.37	*N/A*	0.32	0.37
IEEE 5 GHz decision tree	0.55	0.55	*N/A*	0.49	0.55
IEEE 5 GHz random forest	0.56	0.57	*N/A*	0.51	0.57
Unweighted combi special DT	0.54	0.75	*N/A*	0.67	0.64
Weighted combi special DT	0.55	0.57	*N/A*	0.53	0.56
Unweighted combi special RF	0.53	0.71	*N/A*	0.64	0.62
Weighted combi special RF	0.54	0.59	*N/A*	0.55	0.57
Combi general DT	0.57	0.65	*N/A*	0.58	0.61
Combi general RF	0.73	0.77	*N/A*	0.71	0.75

(c) Sender: Raspberry Pi 4b

Table 4.4: Classifier overview for the *bus* evaluation set (accuracy biased by unbalanced classes)

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds	0.69	0.00	0.29	0.44	0.33
IEEE 2.4 GHz decision tree	0.33	0.25	0.17	0.26	0.25
IEEE 2.4 GHz random forest	0.36	0.22	0.17	0.27	0.25
IEEE 5 GHz decision tree	0.20	0.27	0.18	0.21	0.22
IEEE 5 GHz random forest	0.18	0.30	0.14	0.19	0.21
Unweighted combi special DT	0.07	0.12	0.31	0.14	0.17
Weighted combi special DT	0.13	0.09	0.23	0.15	0.15
Unweighted combi special RF	0.05	0.14	0.29	0.14	0.16
Weighted combi special RF	0.13	0.09	0.20	0.14	0.14
Combi general DT	0.02	0.41	0.11	0.15	0.18
Combi general RF	0.03	0.37	0.17	0.19	0.19

(a) Sender: OnePlus Nord N10 5G

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds	0.78	0.66	0.00	0.69	0.48
IEEE 2.4 GHz decision tree	0.30	0.29	0.14	0.25	0.25
IEEE 2.4 GHz random forest	0.26	0.44	0.09	0.29	0.26
IEEE 5 GHz decision tree	0.18	0.49	0.21	0.34	0.29
IEEE 5 GHz random forest	0.19	0.56	0.26	0.40	0.34
Unweighted combi special DT	0.29	0.51	0.06	0.40	0.29
Weighted combi special DT	0.34	0.51	0.17	0.40	0.34
Unweighted combi special RF	0.26	0.57	0.05	0.44	0.29
Weighted combi special RF	0.33	0.57	0.19	0.45	0.36
Combi general DT	0.17	0.49	0.27	0.36	0.31
Combi general RF	0.13	0.56	0.26	0.40	0.32

(b) Sender: Apple iPhone 6S

Classifier	F_1-**score very close**	F_1-**score close**	F_1-**score safe**	**accuracy***	**macro** F_1-**score**
BLE thresholds only	0.00	0.18	0.63	0.35	0.27
IEEE 2.4 GHz decision tree	0.13	0.24	0.42	0.29	0.26
IEEE 2.4 GHz random forest	0.11	0.19	0.46	0.29	0.25
IEEE 5 GHz decision tree	0.00	0.22	0.33	0.24	0.18
IEEE 5 GHz random forest	0.00	0.36	0.44	0.34	0.27
Unweighted combi special DT	0.00	0.14	0.49	0.27	0.21
Weighted combi special DT	0.00	0.12	0.43	0.27	0.19
Unweighted combi special RF	0.00	0.16	0.55	0.31	0.24
Weighted combi special RF	0.00	0.16	0.47	0.30	0.21
Combi general DT	0.00	0.00	0.43	0.28	0.14
Combi general RF	0.00	0.00	0.43	0.28	0.14

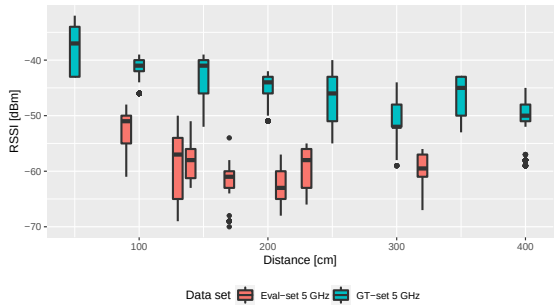
(c) Sender: Raspberry Pi 4b

Table 4.5: Classifier overview for the *meeting* evaluation set (accuracy biased by unbalanced classes)

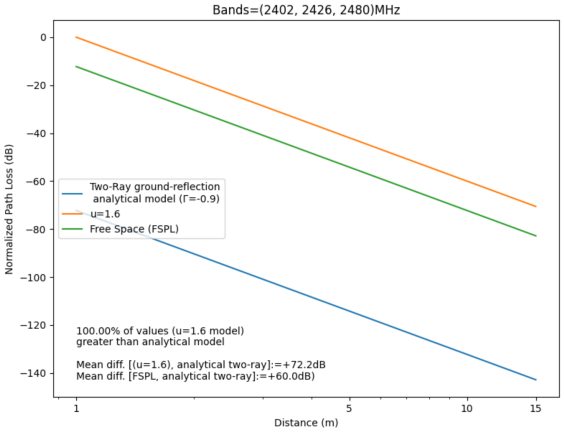
4.8.2 Meeting scenario evaluation set

The same evaluation and data preparation procedure that was used for the *bus* scenarios, was also used for the *meeting* scenarios. We measured several seat scenarios around the *meeting* table. The scenarios are described in more detail in section 3.4.1. We listed the prediction performance results in table 4.5. The results, that can be observed here, confused us first. The classifiers that performed much better than the threshold based BLE classification before, are now doing worse than the BLE approach. With the OnePlus used as sender the macro-f1 score is at 0.33, the next best classifier is the IEEE 2.4 GHz decision tree random forest with a macro F_1-score of 0.25. For the other two devices, the BLE classification also outperforms the other classifiers. A big issue on all three devices seems to be the classification of *very close* distances. But also the other distance classes have very poor metrics.

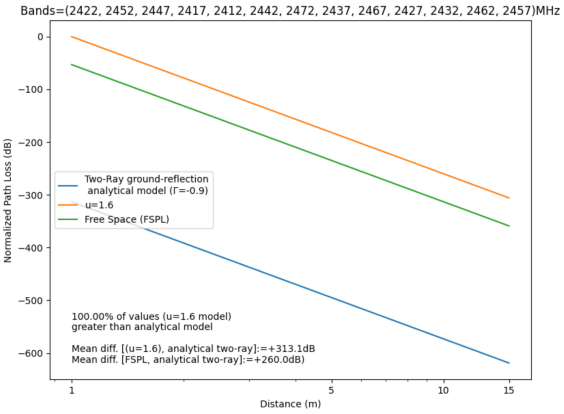
Investigating this poor performance, we observed that most of the samples for all devices have been predicted as safe. To track this issue down, we compared the measured 5 GHz data from the evaluation set and compared it to our ground truth from our *meeting* set. Figure 4.2a shows the result of this comparison. In this plot the RSSI values of the evaluation set are more than 10 dBm weaker for almost each distance, compared to the ground truth values. For 2.4 GHz IEEE signals the situation is comparable abnormal. We consider the experimental setup to be responsible for this phenomenon, because for measuring the *meeting* scenarios with different positions, the devices were directly placed onto a table. Because for measuring the *meeting* scenarios with different positions, the devices were directly place onto a table. Thus, there was the table as a direct obstacle, instead of having a clear diagonal line to the ground from sender side, as we had it in the ground truth measurements with the sender on the very edge of a chair. This then results in a direct (and earlier) signal reflection by the tabletop. We back this up with the analytical Two-Ray Ground-reflection model. For the model we set the sender and receiver height to zero, since both devices were standing on a table surface. The results for all used signal types are visualized in figure 4.2b for the BLE ADV_CHs, figure 4.2c using the 2.4 GHz IEEE 802.11 channels, and figure 4.2d using the 5 GHz channels. By inspecting the results of the model, we can observe a steady growing path loss, without a curve going up and down depending on the distance, as we had it for the ground truth sets. Moreover, the path loss for the two IEEE 802.11 bands is significantly higher than for BLE frequencies, which might explain that the BLE performance result is the only one that is quite comparable to the previous results. The fact that our model was trained on fluctuating signal propagation curves and not on such a steady loss, which occurs with sender and receiver on zero height, makes the poor performance results quite evident.



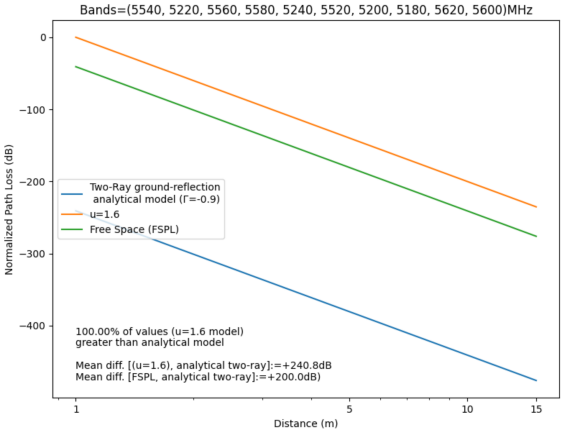
(a) Set comparison between ground truth and *meeting* evaluation set 5 GHz (sender: iPhone)



(b) BLE ADV_CHs Two-Ray Ground-propagation (h=0)



(c) 2.4 GHz Two-Ray Ground-propagation (h=0)



(d) 5 GHz Two-Ray Ground-propagation (h=0)

Figure 4.2: Analysis of signal propagation issues in the *meeting* evaluation set

To put this part of the evaluation in perspective, with this *meeting* scenarios we found a scenario that easily creates a very different signal propagation situation. Such situations can make it hard for every signal strength based measurement approach, to provide accurate results. However, we assume that these zero height situations occur less frequently in everyday life. When they do occur, it is often in situations where the people are known and an appointment has already been recorded. We suspect that our approach will improve the proximity prediction performance in most everyday scenarios. As we have shown this, for example, for a public transport scenario. But of course this is a limitation of our proposed method, that cannot be ignored. We will address this issue later in the future work section.

4.9 Mapping approach evaluation

So far, we have demonstrated that this method can achieve improvements. It is now necessary to check whether a mapping of the signals works in order to be able to use the method based on probe requests in daily operation. Therefore, we implemented approach 3, that was presented section 3.1.2. We will also briefly describe the approach in the following subsection. A direct matching using RSSI values, that we also mentioned, was not possible due to too different signal strength values across the signal types and device types. Briefly described, in our followed approach we try to build traces of MAC addresses, for BLE advertisements and IEEE 802.11 probe requests, in order to identify a device and the MAC addresses it had. Then we compared all the resulting traces for both signal types with their timestamps and matched the devices based on the time window they were active. In other words, when we identified a BLE device that was active from timestamp t_1 until t_2, and we also observed a 802.11 device that was active for more or less the same time, it is likely that this is the same device, and we can map the signals together. The challenging part of this procedure is identifying the traces of a device. We will now briefly explain the procedure for each signal type and will analyse the quality of this identification procedure. Our ground assumption for this is, that all advertisements and probe requests sent out by a device are constantly sniffed by our infrastructure and no packet will be unobserved.

4.9.1 Identifying BLE device traces

The idea we followed for the identification of a BLE device trace was briefly described in 3.1.2, based on the data we measured then, we came up with some improvements. Basically the procedure we are following is that we want to reduce the number of possible MAC addresses, that a device could have after changing, as far as possible. By this we want to achieve the goal, that we can then make a prediction of its new MAC address with a high probability. The best case, of course, is when after the reduction process only one possible next MAC address is left. We used the following procedure to reduce the set size of the most likely next MAC address:

- Splitting all packets into two sets per operating system type, iOS and Android, based on the payload length (described in section 3.6.1).
- Removing all MAC address that sent fewer 10 packets.

- Finding the first time an address was seen and the last time it was seen for each address, in order to filter all other addresses that have been seen before a change of the current address out. Furthermore, we filter out all addresses out that have a starting time less than 30 seconds after the current address was last seen. Then we identify the MAC address that appeared directly after change.
- Calculating the average RSSI of all packets of the current MAC address, and the candidates left after time filtering, to find the candidate with the lowest RSSI delta to the current MAC address.

In most of the cases we analysed in our data, the MAC address seen next by timestamp was the actual new MAC address and it also had the lowest RSSI delta. In some rare cases, however, this time collided with another MAC address, that started broadcasting before a packet with the actual new MAC address of the device was received. To overcome this collision issue, we always used the MAC address with the lowest RSSI delta in our set (of new MAC addresses in the next 30 seconds), as the next address.

Using this procedure we were able to obtain very accurate device traces. We applied this to all data recorded in the *bus* and *train* environment, and on the *meeting* evaluation settings, too. We manually classified the traces and compared them with the traces we found. Our implementation of this approach was able to find almost all BLE traces in our data correctly. The only issue were missing start or end addresses, that were not included in our automatically extracted traces, because in some cases the minimum number of 10 packets was not reached.

4.9.2 Identifying 802.11 probe request device traces

To identify the 802.11 devices and their MAC addresses, we utilized the HT/VHT information available in a probe request frame, as proposed in section 3.1.2. For the devices we used in our measurements it was even easier, and we were able to identify them by the length of the probe request packet. In a preprocessing step we reduced the number of packets, by only keeping packets that were sent with a random MAC address. Moreover, high RSSI values (depending on the environment, e.g, less -80 dBm), were filtered out. Then the resulting set was split into 2.4 GHz probe requests and 5 GHz probe requests, because the capability information varies according to the band used. By filtering these two sets on specific lengths, we were able to extract the packets that have been sent using our sending devices. This procedure was also successful with the data we captured in the *train* setting, were more (unknown) devices were around.

However, with a growing number of devices that send out probe requests, the probability of having devices of the same type and also the probability of devices that send out probe requests of the same length, grows. Then we have to change our approach and perform an individual fingerprinting of the fields in the probe request's frame, such as vendor specific information and the capability fields. Moreover, approaches from the BLE filtering method can be utilized too, such as the RSSI comparison, the RSSI per channel stays very constant in 5 GHz for example, this could definitely be used to improve the procedure. But due to the fact that we have not had data with more devices sending probe requests, we have not evaluated these improvements yet. This is a part that could be done in future work.

When we have a set of traces for both signal types now, we can match them by the times seen. This gets easier when the device type is known. For example, we knew which packet lengths of the IEEE 802.11 probe requests belonged to our iOS device and which to the Android device. Since we also knew this for the BLE traces, the matching of our traces was quite simple. Having such knowledge also for multiple device types, e.g., obtained from further measurements, the mapping can also be improved for scenarios with more devices than we had.

4.9.3 Limitations of the mapping

The approach we are using does have some limitations. The identification and the likelihood of time collisions in the change of BLE MAC addresses increases with more devices are around. It gets even more difficult for non-static scenarios, with moving devices. Then our verification using RSSI deltas, will not work properly any more. With the fingerprinting approach we think that the IEEE 802.11 trace identification is more robust against moving devices. But here, a growing device number, especially many devices of the same type, may cause a less accurate identification. In general, this method has an accuracy that will decrease depending on the number of devices. Moreover, another big issue is the sporadic sending of IEEE 802.11 probe requests. When a device is connected to a network or the Wi-Fi is turned off, no probe requests will be sent. Additionally, even with Wi-Fi turned on, when the screen is off, some devices do not send any probe requests or just very few, sporadically. This will result in much shorter time windows for the IEEE 802.11 traces that will cause issues in matching the traces.

In fact, due to the limitations mentioned above, we suspect that this method, utilizing probe requests, will not work optimally in daily scenarios. Furthermore, there are privacy implications that should not be neglected. We will now discuss these implications and conclude with the overall approach in the discussion chapter.

4.9.4 Privacy considerations

Using an IEEE 802.11 probe request mapping approach incurs a privacy risk. This risk is distinct from and comes on top of to the already mentioned privacy issues of GAEN (section 2.11.4) and probe requests in general (section 2.7.3). As mentioned, probe requests can be used to identify devices, e.g., when they sent queries looking for specific SSIDs in their probe requests. Combining this with the issue of tracking the position of a device in a certain area, using the constantly send advertisements, one might set up a personalized location tracking. Furthermore, it would be possible to obtain the infection status of such a tracked person, by comparing the RPIs that were sniffed to the TEKs published via GAEN. Thus, a major privacy assurance of GAEN, that infections reported in the system stay anonymous, gets broken. The description of having a privacy preserving contact tracing system, is highly conflicted by these facts. To lower the risk, a special IEEE 802.11 frame type, containing less information than a probe request frame, may solve this problem. We will discuss this idea in more detail in the next chapter.

4.10 Summary

In this chapter we presented our evaluation concept in the beginning. Then we presented the classification performance that can be achieved by using specialized single channel classifiers, that used IEEE 802.11 2.4 GHz or 5 GHz signals. On our data these classifiers had a better performance than the approach using BLE thresholds. Afterwards we tried to combine these specialized classifiers and the BLE classifier. Therefore, we used an unweighted and a weighted approach to combine the three signal types. The weighted approach showed improved results with macro f1-scores higher than 0.8 which is a significant improvement compared to the BLE only approach. Next, we trained a decision tree and a random forest using data from all three signal types. The evaluation results using these general classifiers improved further to f1-scores higher than 0.9. Further, we evaluated these models using two other data sets, that were not used for training and evaluation so far. These were the *bus* scenario evaluation set and the *meeting* scenario evaluation set. For the *bus* set our results stayed more than two times better than the BLE only performance. But for the *meeting* scenario evaluation set, the performance of all of our classifiers was worse. We investigated these results and found out that this issue occurred resulted from higher reflections caused by the tabletop the devices were standing on. Then our approach for mapping BLE to IEEE 802.11 signals was evaluated. We obtained good results for static settings, but also identified limitations for moving scenarios. These limitations, in our view, make it difficult to use our approach in practice. We ended with an analysis regarding privacy considerations, that occur when IEEE 802.11 probe requests are mapped to BLE signals and also used in proximity tracing.

5 Conclusion and Future work

Finally, in this last chapter, we will first give a conclusion and discuss our work as a whole. Then we will identify possibilities for future work that can be carried out to improve our approach.

5.1 Conclusion and Discussion

In this thesis we proposed a method to improve the accuracy of the distance estimation used in privacy preserving proximity tracing systems. The proximity tracing systems used during the COVID-19 pandemic, mostly used the signal strength of BLE signals to estimate the proximity. Our own measurements and other studies showed that using BLE signals with thresholds for proximity estimation is a very error-prone approach, that can be heavily influenced by many factors. To overcome these issues we proposed an approach of adding signal strength information of IEEE 802.11 signals, for both bands 2.4 GHz and 5 GHz, to perform a more accurate proximity estimation. To combine these signals we evaluated several machine learning methods and classifier types. We concluded that a general classifier, using a decision tree or random forest trained on all three signal types (attenuation or RSSI and frequency), achieves a significant improvement of the proximity estimation quality, compared to the widely used BLE approach. The significant improvements were successfully evaluated using two of our three evaluations data sets. In the evaluation using the third set, we identified the limitation, that our approach is not robust against signal propagation scenarios that result in very high path-loss. In our case, this was caused by a signal reflecting table-top, where the sender and receiver devices were standing on. More generally, any method that uses signal strength values to estimate the distance between two points has the limitation imposed by the medium itself and its propagation characteristics. Therefore, it is a challenge to use such methods in this field. However, due to the lack of commonly available alternatives, this is the only option that is simple and quick to use, from which the best must be taken. With our approach of combining IEEE 802.11 and BLE signals, we expect that in daily proximity tracing use-cases our approach will outperform the BLE only approach and we are able to improve the accuracy of the distance estimation.

In our study, we used IEEE 802.11 probe request frames, as broadcast messages on IEEE 802.11, in order to combine them with the BLE advertisements. We first assumed that this would be a good approach, because the messages are usually sent out by mobile devices and it would result in an easier deployment of our approach by using that existing signals. Unfortunately, we then realized that there are, on the one hand several limitations using such probe requests. For example there are many cases where these messages are not sent, which result in a difficult mapping of the signal types. On the other hand, probe requests contain information that make it easier to identify a device or user, by sniffing a probe request. The combination of probe requests and BLE advertisements sent out by a proximity tracing system, such as GAEN, will result in an increased risk on the user privacy. To overcome this privacy risk, we propose the usage of a customized data frame. Such a data frame should contain the same payload that is used for the BLE advertisements, so basically the RPI. This new custom frame would also solve the issue of matching BLE signals with IEEE 802.11 signals. However, sending out such a frame type, needs the cooperation from the operating system vendors, that such a features will be integrated in GAEN. On normal application level, there is no permission to inject custom data frames to the wireless interface.

Our general conclusion is that our approach, with the adjustments just mentioned, is a very good extension to the currently used GAEN system. In many everyday scenarios, the combination of IEEE 802.11 and BLE signals would significantly improve proximity estimation. However, we also see various potentials for improvement of our approach in order to reduce the limitations.

5.2 Future work

The most important point is evaluating the usage of custom IEEE 802.11 data frames, for contact tracing. These frames will enable an easy fusion with the data captured via BLE and will reduce the privacy risk that results from using IEEE 802.11 probe requests, as we evaluated it in this work. For the most used smartphone operating systems it needs to be evaluated, what requirements have to be fulfilled for sending and receiving custom IEEE 802.11 broadcast frames. Additionally, there may be firmware requirements for smartphones, in order to capture packages in a not network connected state (monitor mode), that need to be examined. Moreover, the impact on the battery of a smartphone by regularly sending out IEEE 802.11 broadcast messages needs to be assessed. We assume that a constant sending or with sending intervals of 10 seconds, such it is done by BLE advertisements, will drain the battery enormously. Therefore, it can be useful to develop strategies of opportunistic sending, for example that an IEEE

802.11 packet is only sent out, when other devices have been seen via BLE before. A promising way would be, e.g., the use of the trickle algorithm [49].

Next, another very important point is our "table-top limitation" or precisely situations that have difficult signal propagation properties. When it would be possible to detect such situations, a calibration factor could be added, for example, in order to obtain more reliable RSSI values. A detection of such situations may be possible by finding patterns in the signal strength across the three signal types we used. In the evaluation we saw, there was much less path loss that affected BLE than the high path loss affecting IEEE 802.11, in such situations. Such differences between the signal types may help to detect such propagation issues. Therefore, further analysis of the data we already captured and other new scenarios needs to be carried out. If there are promising insights it might be also possible to train a machine model for detecting such situations.

The detection of certain situation, could also be more extended in a way, that also movements and the device orientation can be taken into account for the distance estimation. This idea was also proposed by participants of the NIST machine learning challenge, in order to improve the BLE classifications results. The participants proposed the utilization of gyroscope and IMU sensor data, in order to detect movements and the orientation of a device. This information can then be used by a distance estimations model to use higher or lower thresholds for classification. In our work, we have not evaluated scenarios with moving devices. But we believe that our approach would show weaknesses in such scenarios. This is why we consider it useful to deal with this topic in future work as well. The utilization of data from other sensors built-in sensors in a smartphone seems like a promising approach.

One further point of interest that we would like to review is the improvement of our classification. A simple extension is the usage of more training data from other scenarios, that we have not measured so far. A rather larger adjustment would be to use multiple data points for classification. We currently used only a single sample that contained the three signal types for classification. This means the model has only the signal strength information for one 2.4 GHz channel and one 5 GHz channel. Since contact durations of about 10 minutes are of interest in contact tracking, it would also be possible to enter the data points of such a period as a vector into a model. Then information for various frequencies could be used for classification, which could result in a better classification performance. Such an approach could easily be evaluated on our existing data.

The last issue for future work that we want to mention is the easier applicability of our method for numerous device types without training a random forest or decision tree for each type. It is therefore necessary to check whether it is possible to obtain RSSI-calibration values for a set of devices that allow the use of a general and device-unspecific model.

6 Appendix

6.1 Index of abbreviations

API Application Programming Interface

BLE Bluetooth Low Energy

DP-3T Decentralized Privacy-Preserving Proximity Tracing

GAEN Google/Apple Exposure Notification

GNSS Global Navigation Satellite System

GPS Global Positioning System

IMU Inertial measurement unit

IoT Internet of Things

LOS line of sight

PEPP-PT Pan-European Privacy-Preserving Proximity Tracing

PDU Protocol Data Unit

RMSE Root mean squared error

RSSI Received Signal Strength Indicator

RPI Rolling Proximity Indentifier

TEK Temporary Exposure Key

WHO World Health Organization

Bibliography

[1] Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* (2009), 1–565.

[2] Ieee standard for information technology– telecommunications and information exchange between systemslocal and metropolitan area networks– specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz. *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)* (2013), 1–425.

[3] Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), 1–3534.

[4] Ahmed, N., Michelin, R. A., Xue, W., Ruj, S., Malaney, R., Kanhere, S. S., Seneviratne, A., Hu, W., Janicke, H., and Jha, S. K. A survey of covid-19 contact tracing apps. *IEEE Access 8* (2020), 134577–134601.

[5] Alpaydin, E. *Introduction to machine learning*. MIT press, 2020.

[6] Apple and Google. Exposure notifications api. `https://developers.google.com/android/exposure-notifications/exposure-notifications-api`. (Accessed on 10/06/2020).

[7] Apple and Google. Exposure notifications ble attenuations. `https://developers.google.com/android/exposure-notifications/ble-attenuation-overview`. (Accessed on 10/05/2020).

[8] Apple and Google. Exposure notification reference - bluetooth specification. `https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf`, April 2020. (Accessed on 11/01/2020).

[9] Apple and Google. Exposure notification reference - cryptography specification. `https://blog.google/documents/69/Exposure_Notification_-_Cryptography_Specification_v1.2.1.pdf`, April 2020. (Accessed on 11/01/2020).

[10] Baumgärtner, L., Dmitrienko, A., Freisleben, B., Gruler, A., Höchst, J., Kühlberg, J., Mezini, M., Miettinen, M., Muhamedagic, A., Nguyen, T. D., et al. Mind the gap: Security & privacy risks of contact tracing apps. *arXiv preprint arXiv:2006.05914* (2020).

[11] Bensky, A. *Short-range wireless communication*. Newnes, 2019.

[12] Blaunstein, N., and Christodoulou, C. G. *Radio Propagation and Adaptive Antennas for Wireless Communication Networks: Terrestrial, Atmospheric, and Ionospheric*. John Wiley & Sons, 2014.

[13] Bluetooth Special Interest Group (SIG). Core specifications 5.2. `https://www.bluetooth.com/specifications/bluetooth-core-specification/`. (Accessed on 09/29/2020).

[14] Busvine, D., and Rinke, A. Germany flips to apple-google approach on smartphone contact tracing. `https://reut.rs/2KDdoId`, April 2020. (Accessed on 02/03/2021).

[15] Čabarkapa, D., Grujić, I., and Pavlović, P. Comparative analysis of the bluetooth low-energy indoor positioning systems. In *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)* (2015), IEEE, pp. 76–79.

[16] Carreras, I., Matic, A., Saar, P., and Osmani, V. Comm2sense: Detecting proximity through smartphones. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops* (2012), pp. 253–258.

[17] Castelluccia, C., Bielova, N., Boutet, A., Cunche, M., Lauradoux, C., Le Métayer, D., and Roca, V. Robert: Robust and privacy-preserving proximity tracing.

[18] Chaos Computer Club. 10 requirements for the evaluation of "contact tracing" apps. `https://www.ccc.de/en/updates/2020/contact-tracing-requirements`, April 2020. (Accessed on 02/03/2021).

[19] Chaos Computer Club. Corona-tracing-app: Offener brief an bundeskanzleramt und gesundheitsminister (german). `https://www.ccc.de/de/updates/2020/corona-tracing-app-offener-brief-an-bundeskanzleramt-und-gesundheitsminister`, April 2020. (Accessed on 02/11/2021).

[20] Chen, D., Shin, K. G., Jiang, Y., and Kim, K.-H. Locating and tracking ble beacons with smartphones. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, 2017), CoNEXT '17, Association for Computing Machinery, p. 263–275.

[21] Chowdhury, T. I., Rahman, M. M., Parvez, S., Alam, A. K. M. M., Basher, A., Alam, A., and Rizwan, S. A multi-step approach for rssi-based distance estimation using smartphones. In *2015 International Conference on Networking Systems and Security (NSysS)* (2015), pp. 1–5.

[22] Clark, L., Papalia, A., Carvalho, J. T., Mastrostefano, L., and Krishnamachari, B. Inter-mobile-device distance estimation using network localization algorithms for digital contact logging applications. *Smart Health 19* (2021), 100168.

[23] Corman, V., Bleicker, T., Brünink, S., Drosten, C., Zambon, M., Organization, W. H., et al. Diagnostic detection of wuhan coronavirus 2019 by real-time rt-pcr. *Geneva: World Health Organization, January 13* (2020).

[24] Coronaviridae Study Group of the International Committee on Taxonomy of Viruses. The species severe acute respiratory syndrome-related coronavirus: classifying 2019-ncov and naming it sars-cov-2. *Nature Microbiology 5*, 4 (2020), 536.

[25] Darrell Etherington, N. L. Apple and google update joint coronavirus tracing tech to improve user privacy and developer flexibility. `https://techcrunch.com/2020/04/24/apple-and-google-update-joint-coronavirus-tracing-tech-to-improve-user-privacy-and-developer-flexibility/`, April 2020. (Accessed on 02/11/2021).

[26] Dong, E., Du, H., and Gardner, L. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases 20*, 5 (2020),

533–534.

[27] ECDC. Timeline of ecdc's reponse to covid-19. `https://www.ecdc.europa.eu/en/covid-19/timeline-ecdc-response`. (Accessed on 12/29/2020).

[28] Faragher, R., and Harle, R. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)* (2014), pp. 201–210.

[29] Fraser, C., Abeler-Dörner, L., Ferretti, L., Parker, M., Kendall, M., and Bonsall, D. Digital contact tracing: comparing the capabilities of centralised and decentralised data architectures to effectively suppress the covid-19 epidemic whilst maximising freedom of movement and maintaining privacy. *University of Oxford* (2020).

[30] Freudiger, J. How talkative is your mobile device? an experimental study of wi-fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (2015), pp. 1–6.

[31] Gast, M. *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.", 2005.

[32] Gast, M. *802.11 n: a survival guide*. " O'Reilly Media, Inc.", 2012.

[33] Gast, M. S. *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond*. " O'Reilly Media, Inc.", 2013.

[34] Gentner, C., Günther, D., and Kindt, P. H. Identifying the ble advertising channel for reliable distance estimation on smartphones. *arXiv preprint arXiv:2006.09099* (2020).

[35] Ghamari, M., Villeneuve, E., Soltanpur, C., Khangosstar, J., Janko, B., Sherratt, R. S., and Harwin, W. Detailed examination of a packet collision model for bluetooth low energy advertising mode. *IEEE Access 6* (2018), 46066–46073.

[36] Government Technology Agency Singapore. opentrace-calibration. `https://github.com/opentrace-community/opentrace-calibration/blob/master/Trial%20Methodologies.md`. (Accessed on 09/29/2020).

[37] Gu, X., Wu, W., Gu, X., Ling, Z., Yang, M., and Song, A. Probe request based device identification attack and defense. *Sensors 20*, 16 (2020), 4620.

[38] He, T., and Printz, M. A 2-stage classifier for contact detection with bluetoothle and ins signals. *NIST TC4TL Challenge* (2020).

[39] Hoepman, J.-H. A critique of the google apple exposure notification (gaen) framework. *arXiv preprint arXiv:2012.05097* (2020).

[40] Hsu, J. Can ai make bluetooth contact tracing better? `https://spectrum.ieee.org/the-human-os/artificial-intelligence/machine-learning/ai-bluetooth-contact-tracing`, September 2020. (Accessed on 04/05/2021).

[41] Illmer, A. Singapore reveals covid privacy data available to police - bbc news. `https://www.bbc.com/news/world-asia-55541001`, January 2020. (Accessed on 01/06/2021).

[42] James Larus, e. a. Joint statement on contact tracing: Date 19th april 2020. `https://drive.google.com/file/d/1OQg2dxPu-x-RZzETlpV3lFa259Nrpk1J/vie`, April 2020. (Accessed on 02/10/2021).

[43] Kuhn, C., Beck, M., and Strufe, T. Covid notions: Towards formal definitions – and documented understanding – of privacy goals and claimed protection in proximity-tracing services. *Online Social Networks and Media 22* (2021), 100125.

[44] Leith, D., and Farrell, S. Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps.

[45] Leith, D. J., and Farrell, S. Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection. *ACM SIGCOMM Computer Communication Review 50*, 4 (2020), 66–74.

[46] Leith, D. J., and Farrell, S. Measurement-based evaluation of google/apple exposure notification api for proximity detection in a light-rail tram. *PLOS ONE 15*, 9 (09 2020), 1–16.

[47] Leith, D. J., and Farrell, S. Measurement-based evaluation of google/apple exposure notification api for proximity detection in a light-rail tram. *PLOS ONE 15*, 9 (09 2020), 1–16.

[48] Leith, D. J., and Farrell, S. Measurement-based evaluation of google/apple exposure notification api for proximity detection in a commuter bus. *Plos one 16*, 4 (2021), e0250826.

[49] Levis, P., Patel, N., Culler, D., and Shenker, S. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor

networks. In *Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation* (2004), vol. 25.

[50] Lewis, D. Why many countries failed at covid contact-tracing-but some got it right. *Nature 588*, 7838 (2020), 384–387.

[51] Liu, S., Jiang, Y., and Striegel, A. Face-to-face proximity estimation using bluetooth on smartphones. *IEEE Transactions on Mobile Computing 13*, 4 (Apr. 2014), 811–823.

[52] Lucivero, F., Hallowell, N., Johnson, S., Prainsack, B., Samuel, G., and Sharon, T. Covid-19 and contact tracing apps: Ethical challenges for a social experiment on a global scale. *Journal of bioethical inquiry 17*, 4 (2020), 835–839.

[53] Martin, T., Karopoulos, G., Hernández-Ramos, J. L., Kambourakis, G., and Nai Fovino, I. Demystifying covid-19 digital contact tracing: A survey on frameworks and mobile apps. *Wireless Communications and Mobile Computing 2020* (2020).

[54] Morley, J., Cowls, J., Taddeo, M., and Floridi, L. Ethical guidelines for covid-19 tracing apps, 2020.

[55] Ng, A. Nuts and bolts of building ai applications using deep learning. *NIPS Keynote Talk* (2016).

[56] Nikoukar, A., Abboud, M., Samadi, B., Güneş, M., and Dezfouli, B. Empirical analysis and modeling of bluetooth low-energy (ble) advertisement channels. In *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)* (2018), IEEE, pp. 1–6.

[57] O'Neill, P. H. No, coronavirus apps don't need 60% adoption to be effective. `https://www.technologyreview.com/2020/06/05/1002775/covid-apps-effective-at-less-than-60-percent-download/`, June 2020. (Accessed on 11/02/2020).

[58] Palaghias, N., Hoseinitabatabaei, S. A., Nati, M., Gluhak, A., and Moessner, K. Accurate detection of real-world social interactions with smartphones. In *2015 IEEE International Conference on Communications (ICC)* (2015), pp. 579–585.

[59] PEPP-PT Consortium. Data protection and information security architecture, illustrated on german implementation. `https://github.com/pepp-pt/pepp-pt-documentation/blob/master/10-data-protection/PEPP-PT-`

data-protection-information-security-architecture-Germany.pdf, April 2020. (Accessed on 02/10/2021).

[60] RKI. Kennzahlen zur corona-warn-app [german]. https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/WarnApp/Archiv_Kennzahlen/Kennzahlen_05022021.pdf?__blob=publicationFile, February 2021. (Accessed on 02/11/2021).

[61] Samarati, P., and Sweeney, L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.

[62] SAP and Telekom. How does the corona-warn-app identify an increased risk? https://github.com/corona-warn-app/cwa-documentation/blob/master/cwa-risk-assessment.md, July 2020. (Accessed on 11/02/2020).

[63] Specht, C., Dabrowski, P., Pawelski, J., Specht, M., and Szot, T. Comparative analysis of positioning accuracy of gnss receivers of samsung galaxy smartphones in marine dynamic measurements. *Advances in Space Research 63*, 9 (2019), 3018–3028.

[64] Torkamandi, P., Kärkkäinen, L., and Ott, J. An online method for estimating the wireless device count via privacy-preserving wi-fi fingerprinting. In *Passive and Active Measurement* (Cham, 2021), O. Hohlfeld, A. Lutu, and D. Levin, Eds., Springer International Publishing, pp. 406–423.

[65] Troncoso, C., Payer, M., Hubaux, J., Salathé, M., Larus, J., Lueks, W., Stadler, T., Pyrgelis, A., Antonioli, D., Barman, L., et al. Decentralized privacy-preserving proximity tracing. *IEEE Data Engineering Bulletin 43*, 2 (2020), 36–66.

[66] Tsiflikiotis, Antonis. Two-ray-ground-reflection-model. https://github.com/atsiflikiotis/Two-ray-ground-reflection-model, May 2020. (Accessed on 05/23/2021).

[67] Tsimbalo, E., Fafoutis, X., Mellios, E., Haghighi, M., Tan, B., Hilton, G., Piechocki, R., and Craddock, I. Mitigating packet loss in connectionless bluetooth low energy. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)* (2015), IEEE, pp. 291–296.

[68] Vanhoef, M., Matte, C., Cunche, M., Cardoso, L. S., and Piessens, F. Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (New York, NY, USA, 2016), ASIA CCS '16, Association for Computing Machinery, p. 413–424.

[69] Varshney, V., Goel, R. K., and Qadeer, M. A. Indoor positioning system using wi-fi bluetooth low energy technology. In *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)* (2016), pp. 1–6.

[70] Wang, C., Horby, P. W., Hayden, F. G., and Gao, G. F. A novel coronavirus outbreak of global health concern. *The Lancet 395*, 10223 (2020), 470–473.

[71] WHO. Coronavirus disease (covid-19): How is it transmitted? https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-covid-19-how-is-it-transmitted. (Accessed on 12/30/2020).

[72] WHO. Critical preparedness, readiness and response actions for covid-19. https://www.who.int/publications/i/item/critical-preparedness-readiness-and-response-actions-for-covid-19. (Accessed on 12/29/2020).

[73] WHO. Go.data. https://www.who.int/godata/. (Accessed on 01/04/2021).

[74] WHO. Listings of who's response to covid-19. https://www.who.int/news/item/29-06-2020-covidtimeline. (Accessed on 12/29/2020).

[75] WHO. Contact tracing in the context of covid-19. https://www.who.int/publications/i/item/contact-tracing-in-the-context-of-covid-19, May 2020. (Accessed on 12/30/2020).

[76] WHO. Digital tools for covid-19 contact tracing. https://www.who.int/publications/i/item/WHO-2019-nCoV-Contact_Tracing-Tools_Annex-2020.1, June 2020. (Accessed on 01/03/2021).

[77] WHO. Mission summary: Who field visit to wuhan, china 20-21 january 2020. https://www.who.int/china/news/detail/22-01-2020-field-visit-wuhan-china-jan-2020, January 2020. (Accessed on 12/29/2020).

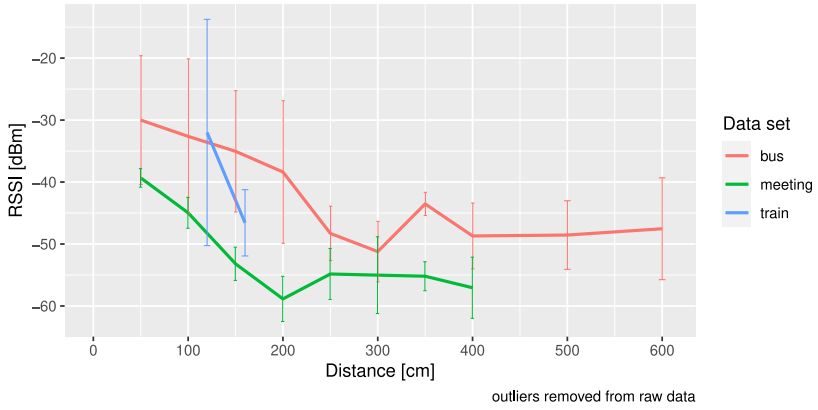
[78] WHO. Novel coronavirus – china. https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/, January 2020. (Accessed on 12/29/2020).

[79] WHO. Novel coronavirus(2019-ncov) situation report - 10. https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200130-sitrep-10-ncov.pdf?sfvrsn=d0b2e480_2, January 2020. (Accessed on 12/29/2020).

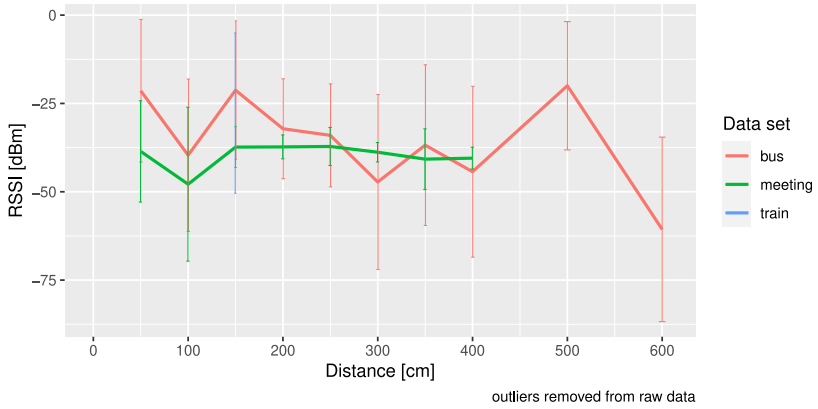
[80] WHO. Who director-general's opening remarks at the media briefing on covid-19 - 11 march 2020. https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020, March 2020. (Accessed on 12/29/2020).

[81] Zhao, Q., Wen, H., Lin, Z., Xuan, D., and Shroff, N. On the accuracy of measured proximity of bluetooth-based contact tracing apps. https://web.cse.ohio-state.edu/~lin.3021/file/SECURECOMM20b.pdf, 2020. (Accessed on 10/13/2020).

6.2 Figures



(a) iPhone 6S



(b) Raspberry Pi 4b

Figure 6.1: Environment comparison IEEE 802.11 2.4 GHz

6.3 Listings

```
from ortools.sat.python import cp_model

# initialize variables
c_2 = model.NewIntVar(-90, 90, 't_2')
sep_2 = model.NewIntVar(-80, 0, 'sep_2')
sym_2 = model.NewIntVar(-80, 0, 'sym_2')

# set constraints
model.Add(-11-(-58-c_2) < 55) # 0.5m
model.Add(-11-(-54-c_2) < 55) # 1m
model.Add(-11-(-64-c_2) <= 55)  # 1.5m
model.Add(-11-(sep_2-c_2) > 55)  # seperate VC from C
model.Add(-11-(-66-c_2) < 63) # 2m
model.Add(-11-(-64-c_2) < 63) # 2.5m
model.Add(-11-(-61-c_2) <= 63) # 3m
model.Add(-11-(sym_2-c_1) > 63) # >3m

# optimization target
model.Maximize(sym_2+sep_2)
```

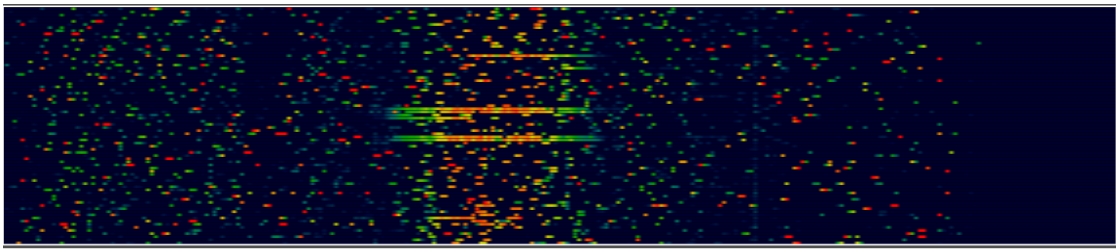
Listing 6.1: Python script used for calibrating OnePlus attenuation

6.4 Measurement issues in office data-set

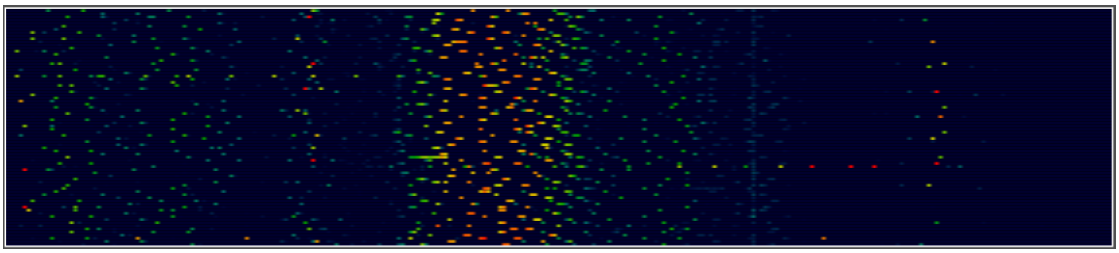
The office set was measured first, it is a larger office room of the distributed systems group at Osnabrück University. The measurements have been carried out during the pandemic, where most people worked from home. So on that floor of the building there were almost no other signals or devices around. On other floors in the building are a medical office and a law firm. The medical office is directly on the floor above and it was working normally to that time. The measurement lane used in the office was already shown in figure 3.3. All measurements on the measurement lane were carried out for d from $50cm$ up to $450cm$. Only ground truth data was collected in this room. The measurements in this environment have been carried out very early, at that point the monitor mode firmware was not compatible with the newest Kernel version (Version $>= 5.x$), and a compilation on the *Raspberry Pi OS* failed. To overcome this issue a Raspberry 3b was used for capturing IEEE 802.11 probe requests in this environment running Kali Linux with a $4.x$ Kernel. All in all, the measurements in this environment helped a lot to improve and fine tune the measurement setup. When most of the data was captured in this setting, the firmware patch was successfully compiled and operational on the Pi4b. Another problem during the measurements was regular signal noise, that was probably coming from medical devices from the medical office on the next floor.

Investigations and package loss measurements on IEEE 802.11 2.4 GHz showed, that there were time-slots in the morning around 8 A.M. and in the evening around 6 P.M., where the package loss was at 100% for 30 minutes up to one hour. The package loss measurement was done by setting up an 2.4 GHz WiFi network, connecting a Raspberry Pi as client to it. On the Pi the software *smokeping*[1] was installed, with a monitoring job targeting the IP address of the device hosting the network. With having the times when no data-transmission was possible identified, spectrum measurements were performed. The spectrum measurement showed that there was significantly more usage at high power rates on the 2.4 GHz band at the identified time-slots. The spectrum plot in figure 6.2a showed the spectrum at time of failure and figure 6.2b the spectrum at a normal state when measurements were carried out. A look into the tool wavemon[2] also showed retry rates about 100% at these time slots. But the connection to the wireless network stayed active, it took about 30 minutes up to one hour, until the transmission was running again. A manual reconnect in that time-slot helps to fix the issue faster. Moreover, the spectrum measurements showed that the higher noise level is only about 5 minutes present. We assume that carrier sense mechanisms are disturbed by this burst of

[1] `https://oss.oetiker.ch/smokeping/`
[2] `https://github.com/uoaerg/wavemon`



(a) At the moment of disturbance



(b) Normal situation

Figure 6.2: Screenshots from spectools spectrum view for 2.4 GHz band in the office environment (Frequency on X-axis, Time on Y-axis)

noise, and the interface needs some time to mitigate with this issue. The concrete source of this disturbance was not found so far. Due to time synchronization issues in some measurement iterations it is not possible to simply cut out the data samples for the certain time windows where the noise level was usually higher. Moreover, because of the device inconsistency and the noise, the measurement set was dropped completely and another test was measured in another room.

Statement of Authorship

I hereby confirm that the work presented in this master thesis has been performed and interpreted solely by myself except where explicitly identified to the contrary. I declare that I have used no other sources and aids other than those indicated. This work has not been submitted elsewhere in any other form for the fulfilment of any other degree or qualification.

Osnabrück, May 25, 2021, ...
Eric Lanfer